AFRL-IF-RS-TR-2005-381
Final Technical Report
November 2005

CONSTRAINTS AND APPROACHES FOR DISTRIBUTED MOBILE AD-HOC NETWORK SECURITY

SUNY Institute of Technology

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-381 has been reviewed and is approved for publication.

APPROVED: /s/

ANDREW J. KARAM
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	NOVEMBER 2005	Final Nov 02 – Nov 03

4. TITLE AND SUBTITLE
CONSTRAINTS AND APPROACHES FOR DISTRIBUTED MOBILE AD-HOC NETWORK SECURITY

5. FUNDING NUMBERS
C - F30602-03-2-0012
PE - 61102F
PR - 2301
TA - 01
WU - 06

6. AUTHOR(S)
Patrick W. Fitzgibbons,
Digen Das, and
Larry J. Hash

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SUNY Institute of Technology
PO Box 3050
Utica New York 13504-3050

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFGB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-381

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Andrew J. Karam/IFGB/(315) 330-7290/ Andrew.Karam@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
This document describes our SAFEMITS network constraints and key management approaches research for year 2. As a first step SUNYIT research team has researched mobile ad-hoc network technology and the unique communications environment in which it will be deployed. We have identified the requirements specific to our problem of providing key management for confidentially and group-level authentication. We have also identified constraints, particularly energy consumption that render this problem difficult.

14. SUBJECT TERMS
SAFEMITS, Constraints, Ad-Hoc Network Security

15. NUMBER OF PAGES
131

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Executive Summary 1

1 Introduction 6

2 Background 8
2.1 SAFEMITS Node Technology 8
2.1.1 SAFEMITS Node Hardware 8
2.1.2 Software 12
2.2 SAFEMITS Network Missions 13
2.2.1 Perimeter Defense or Area Denial 13
2.2.2 Remote Surveillance 13
2.3 SAFEMITS Network Architecture 13
2.3.1 Environment 14
2.3.2 Data Types 14
2.3.3 Communications Architecture 14
2.4 Concept of Operations 18
2.4.1 Manufacture 19
2.4.2 Depot Storage 20
2.4.3 Pre-Deployment 20
2.4.4 Deployment 20
2.4.5 Mission Completion 21
2.5 Environment 21

3 Requirements 22
3.1 Confidentiality 22
3.2 Authenticity 22
3.3 Integrity 22
3.4 Freshness 22
3.5 Scalability 23
3.6 Availability 23
3.7 Accessibility 24
3.8 Self-Organization 24
3.9 Flexibility 25

4 Constraints 26
4.1 SAFEMITS Node Constraints 26
4.1.1 Battery Power/Energy 26
4.1.2 Rechargeability 28
4.1.3 Sleep Patterns 28
4.1.4 Transmission Range 28
4.1.5 Memory 29
4.1.6 Location Communications 30
4.1.7 Tamper Protection 30
4.1.8 Time 30
4.1.9 Unattended Operations 31
4.2 Networking Constraints 31
4.2.1 Ad hoc Networking 31
4.2.2 Limited Pre-Configuration 31
4.2.3 Data Rate/Packet Size 31
4.2.4 Channel error rate 31

4.2.5 Intermittent connectivity 32
4.2.6 Unreliable communications 32
4.2.7 Latency 32
4.2.8 Unicast vs. multicast 32
4.2.9 Unidirectional Communications 32
4.2.10 Isolated subgroups 33
4.2.11 Frequent Routing Changes 33
4.2.12 Population Density 33
4.2.13 Unknown Recipients 33

5 Keying Protocols 34
5.1 Background 34
5.1.1 Key Establishment Steps 34
5.1.2 Basic Keying Techniques 35
5.1.3 Energy Consumption of Keying Primitives 36
5.2 Pre-deployed Keying 43
5.2.1 Network-Wide Pre-deployed Keying 43
5.2.2 Node-Specific Pre-deployed Keying 43
5.2.3 J-Secure Pre-Deployed Keying 45
5.3 Arbitrated Protocols 46
5.3.1 Traditional Key Distribution Center-Based Methods 47
5.3.2 Symmetric Key Certificate-Based Keying 54
5.3.3 Identity-Based Symmetric Keying 58
5.3.4 Arbitrated Group Keying Protocols 62
5.3.5 Energy Consumption Shifting Key Establishment Protocols 67
5.3.6 Public Key Based Kerberos Protocols 76
5.4 Self-Enforcing Autonomous Keying Protocols 77
5.4.1 Pairwise Asymmetric Keying 77
5.4.2 Group Keying Protocols 80
5.4.3 Attribute-Based Keying 93
5.5 Preliminary Techniques Comparison 94

6 Network-wide Approaches 99
6.1 SAFEMITS Network Simulation 99
6.2 Integrating Key Establishment with Network Self-Organization 100
6.3 Group Determination 101
6.4 Hybrid Approaches 103
6.4.1 Pairwise and Group Diffie-Hellman Hybrids 103
6.4.2 Pairwise and Burmester-Desmedt Hybrids 105
6.4.3 Pairwise and Simple Key Distribution Center Hybrids 106
6.4.4 Comparison of Approaches 107
6.5 Energy-Aware Approaches 111
6.5.1 Key Protocol Roles 111
6.5.2 Parasite Protocol 112
6.5.3 Time Varying Approaches 113
6.6 Specialization 113

7 Next Steps 115

References 117

Acronyms 122

List of Figures

Figure 1 - Pairwise and Group Connections, Communications Range = 40 Meters 2
Figure 2 - Power Dissipation for Selected Embedded Processors 9
Figure 3 - SAFEMITS Network Communications Layers 15
Figure 4 - Cluster Algorithm-based Routing 17
Figure 5 - Data Fusion within a SAFEMITS Network 18
Figure 6 - Sample Concept of Operations for SAFEMITS Networking 19
Figure 7 - Establishing Keys Between Small Groups vs. the Entire SAFEMITS Network 25
Figure 8 - SAFEMITS Network Keying Protocols 36
Figure 9 - Kerberos V (modified for DSN use) 49
Figure 10 - Otway-Rees Protocol (modified for DSN use) 52
Figure 11 - Symmetric Certificate Protocol (modified for DSN use) 56
Figure 12 - Identity Based Symmetric Keying Protocol 60
Figure 13 - Logical Key Hierarchy (LKH) 65
Figure 14 - A One-Way Function Tree (OFT) 66
Figure 15 - Rich Uncle Keying Protocol 69
Figure 16 – Pairwise Public Key Based Protocol (PK-TPP) 78
Figure 17 - SAFEMITS August '03 Planned Group A SAFEMITS Positions 100
Figure 18 - Singly-Hop-Connected Groups with Communications Range of 60 Meters 102
Figure 19 - Star Group with Communications Range of 60 Meters 103
Figure 20 - Pairwise and Group Connections, Communications Range = 30 Meters 110
Figure 21 - Pairwise and Group Connections, Communications Range = 35 Meters 110
Figure 22 - Pairwise and Group Connections, Communications Range = 40 Meters 111
Figure 23 - Pairwise and Group Connections, Communications Range = 45 Meters 111
Figure 24 - Routing to Node 1 Plot 112
Figure 25 - Pairwise and Connections, Communications Range = 35 Meters 113

List of Tables

Table 1 - Hybrid Keying Average Energy Consumption, Per Transmission Power Control....3
Table 2 - Hybrid Keying Average Energy Consumption, No Transmit Power Control....3
Table 3 - Battery Charge and Power....11
Table 4 - Computation Time and Energy Consumption for 128-bit Multiply Result....27
Table 5 - AES and SHA-1 Computational Energy Consumption Estimates....27
Table 6 – Computational Energy Costs for Public Key Authentication Algorithms....40
Table 7 - AES Energy Consumption Estimates....41
Table 8 - SHA-1 and MD5 Energy Consumption....42
Table 9 - HMAC Energy Consumption Estimates for a 1024-bit Message....42
Table 10 - Total Number of Keys Required for Node-Specific Pre-deployed Keying....44
Table 11 – Storage Requirements for Node-Specific Pre-Deployed Keying....45
Table 12 – Storage Requirements for J-Secure Pre-Deployed Keying....46
Table 13 – Kerberos Protocol Computational Energy Consumption....50
Table 14 – Kerberos Protocol Multi-hop Total SAFEMITS Node Communication Energy Consumption....50
Table 15 – Otway-Rees Protocol Computational Energy Consumption....53
Table 16 - Otway-Rees Protocol Multi-Hop Communication Energy Consumption....53
Table 17 - Update Message Energy Consumption Required to Add 12 Nodes....54
Table 18 - Energy Consumption per Node for Symmetric Key Certificate Protocol....57
Table 19 - Symmetric Key Certificate Protocol Multi-hop Communication Energy Cost....58
Table 20 - Identity-Based Symmetric Keying Computational Energy Consumption....61
Table 21 – Group Kerberos Protocol Computational Energy Consumption (group size = 6)....64
Table 22 – Group Kerberos Protocol Multi-hop Communication Energy Consumption (group size = 6)....64
Table 23 - Rich Uncle Energy Consumption for MIPS R4000 with WINS Communications....71
Table 24 - Rich Uncle Energy Consumption for MIPS R4000 with MuRF Communications....72
Table 25 - Rich Uncle Energy Consumption for Dragonball with MuRF Communications....72
Table 26 - Relative Energy Consumption for Various Rich Uncle Scenarios....73
Table 27 - Relative Energy Consumption for (Modified) Rich Uncle Scenarios....75
Table 28 – Average Energy Consumption for (Modified) Rich Uncle (6 key-pairs)....75
Table 29 - Energy Consumption of Pairwise Public Key Protocol....80
Table 30 - Energy Consumption of (Elected) Simple Key Distribution Center (Using Rounds 2 and Round 5....83

Table 31 - Energy Consumption of (Elected) Simple Key Distribution Center (with Rounds 2' and 5')....84
Table 32 - Energy cost of A-GDH.3 including certificates with unicast messages only....86
Table 33 - Energy cost of A-GDH.3 including certificates with both unicast and multicast messages....87
Table 34 - Energy Consumption of Authenticated Burmester-Desmedt including certificate transport with unicast messages only....89
Table 35 - Energy Consumption of Authenticated Burmester-Desmedt including certificate transport with unicast and multicast messages....90
Table 36 - Energy Consumption of the Just-Vaudenary Protocol including certificates with unicast messages only....92
Table 37 - Energy Consumption of the Just-Vaudenary Protocol including certificates with both multicast and unicast messages....93
Table 38 - Comparison of Key Establishment Techniques....95
Table 39 - Benefits of Secret-key-Based MAC Authentication for Pairwise RSA....96
Table 40 - Comparison of Pairwise RSA and Rich Uncle Energy Consumption....97
Table 41 - Comparison of Pairwise RSA and Group Keying using Unicast Messages....97
Table 42 - Comparison of Pairwise RSA and Group Keying using Multicast Messages....98
Table 43 – Pairwise-GDH Energy Consumption, Per Transmission Power Control....104
Table 44 – Pairwise-GDH Energy Consumption, No Transmit Power Control....105
Table 45 – Pairwise-BD Energy Consumption, Per Transmission Power Control....105
Table 46 – Pairwise-BD Energy Consumption, No Transmit Power Control....106
Table 47 – Pairwise-SKDC Energy Consumption, Per Transmission Power Control....106
Table 48 – Pairwise-SKDC Energy Consumption, No Transmit Power Control....107
Table 49 - Hybrid Keying Average Energy Consumption, Per Transmission Power Control....108
Table 50 - Hybrid Keying Average Energy Consumption, No Transmit Power Control....108
Table 51 – Maximum Energy Consumption, Per Transmission Power Control....109
Table 52 - Maximum Energy Consumption, No Transmit Power Control....109

Executive Summary

Confidentiality, integrity, and authentication services are critical to preventing an adversary from compromising the security of a distributed SAFEMITS network. Key management is likewise critical to establishing the keys necessary to provide this protection. However, providing key management is difficult due to the ad hoc nature, intermittent connectivity, and resource limitations of the SAFEMITS network environment. As part of the SAFEMITS program, SUNYIT research team is addressing this problem by identifying and developing cryptographic protocols and mechanisms that efficiently provide key management security support services.

This document describes our SAFEMITS network constraints and key management approaches research for year 2. As a first step, SUNYIT research team has researched mobile ad-hoc network technology and the unique communications environment in which it will be deployed. We have identified the requirements specific to our problem of providing key management for confidentiality and group-level authentication. We have also identified constraints, particularly energy consumption, that render this problem difficult.

SUNYIT research team has developed novel key management protocols specifically designed for the distributed SAFEMITS network environment, including Identity-Based Symmetric Keying and Rich Uncle. We have analyzed both existing and SUNYIT research team-developed keying protocols for their suitability at satisfying identified requirements while overcoming battlefield energy constraints. Our research has focused heavily on key management energy consumption, evaluating protocols based on total system, average SAFEMITS node, and individual SAFEMITS node energy consumption.

We examined a number of secret-key-based protocols, determining some to be suitable for SAFEMITS networks but all of the protocols have flexibility limitations. Secret-key-based protocols are generally energy-efficient, using encryption and hashing algorithms that consume relatively little energy. Security of secret-key-based protocols is generally determined by the granularity of established keys, which vary widely for the protocols described herein. During our examination of these protocols we noted that some of these protocols are not sufficiently flexible for use in battlefield SAFEMITS network, since they cannot efficiently handle unanticipated additions of SAFEMITS nodes to the network. Our Identity-Based Symmetric Keying protocol and the less efficient Symmetric Key Certificate Based Protocol are well suited for *certain* mobile ad-hoc networks, establishing granular keys while consuming relatively little energy.

However, all of the secure secret-key-based protocols use special nodes that operate as Key Distribution Centers (or Translators). The SAFEMITS nodes communicate with these centers exchanging information as part of the key establishment process. Since these special nodes are expected to make up less than 1% of the SAFEMITS network's node population, they can only support a very limited number of neighboring SAFEMITS nodes until the SAFEMITS network's *routing infrastructure* is sufficiently well established.

We analyzed public key-based key establishment protocols and determined their flexibility and scalability offer significant advantages in meeting distributed SAFEMITS network needs. Public key-based protocols will establish keys between pairs and small groups of neighboring SAFEMITS nodes within the multi-hop-connected network. The public key algorithms used in these protocols consume a great deal of computational and communications energy, however. We show that group keying can reduce key management energy consumption among groups of singly-hop-connected nodes as small as six. If the SAFEMITS network has multicast communications capabilities, we can further reduce communications energy consumption by using group keying protocols such as Burmester-Desmedt. Alternatively, our public key-based Rich Uncle protocol can offload significant

cryptographic computations, and thus energy consumption, from energy-limited SAFEMITS nodes to energy-endowed "super" nodes.

Our examination of keying protocols has revealed that a single keying protocol will not be optimal for all SAFEMITS network topologies, densities, sizes, and scenarios. Protocols such as Identity-Based Symmetric Keying and Rich Uncle have limited application until the network's *routing infrastructure* has been sufficiently well established. Individually other protocols such as the public-key group and pairwise keying protocols consume too much energy. For *significant* SAFEMITS networks, a mix of public key-based protocols, including pairwise, group keying, and distribution keying, provide an energy-efficiency superior to using just a single protocol.

We have developed group determination algorithms and hybrid key management protocols to improve the energy efficiency of key management. The group determination algorithms find the largest non-overlapping singly-hop-connected and star groups within a given SAFEMITS network field such as those shown in Figure 1. A singly-hop-connected group is a collection of SAFEMITS nodes that can each transmit and receive to every other group member. A star group is as a collection of SAFEMITS nodes that can each transmit and receive to a single "leader" node. The hybrid key management protocols perform various group keying protocols using either singly-hop-connected or star groups, and pairwise keying protocols for any remaining SAFEMITS nodes in the field.

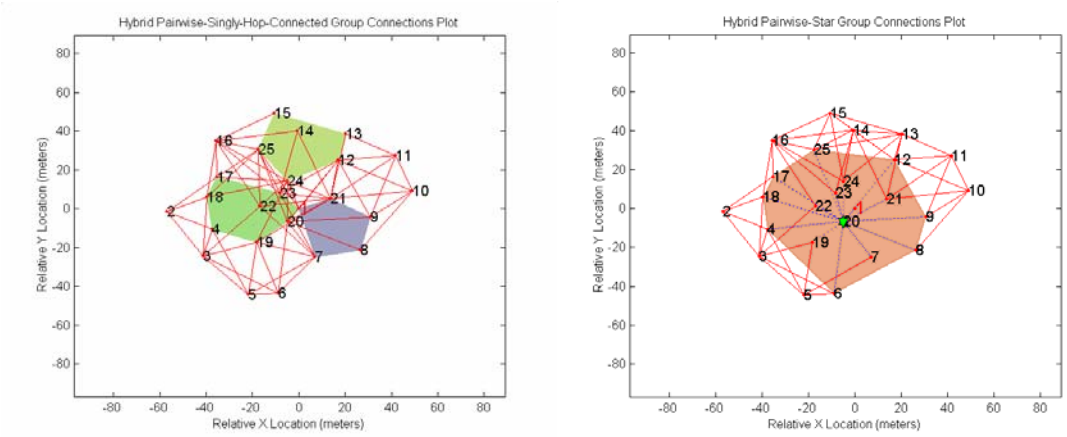

Figure 1 - Pairwise and Group Connections, Communications Range = 40 Meters

We have developed and analyzed a MATLAB-based simulation to assess the performance of our developed algorithms and protocols. Our simulation is based on a routing determination protocol simulation developed by Dr. Diane Mills and Melissa Chevalier of Sanders, a Lockheed Martin Company. The Group A SAFEMITS positions for the SAFEMITS August '00 experiment were used as a baseline for examining the performance of our algorithms and protocols. Different communications ranges and different transmit power control methodologies were simulated for each of the different hybrid approaches.

Our simulation-based analysis demonstrates that hybrid key management protocols provide significant advantages in performing key management. Table 1 compares the average per node key management energy consumption for the pairwise-only baseline with three dual-protocol hybrid schemes when the radiated RF transmission power control is controllable on a per transmission basis. For the majority of communications ranges, the dual-protocol hybrid schemes are significantly more energy-efficient, with the Pairwise-Simple Key Distribution Center (SKDC) hybrid being most efficient.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)			
	Pairwise Only	Pairwise-GDH Hybrid	Pairwise-BD Hybrid	Pairwise-SKDC Hybrid
30	.55	*.58*	.42	**.36**
35	.82	*.87*	.59	**.52**
40	*1.26*	1.21	.79	**.69**
45	*1.73*	1.62	1.05	**.77**
50	*2.28*	1.95	1.30	**.69**
60	*4.42*	3.39	2.28	**.50**
70	*6.63*	4.78	3.32	**.50**
80	*9.67*	6.11	4.29	**.50**
90	*13.42*	6.53	5.33	**.50**

Table 1 - Hybrid Keying Average Energy Consumption, Per Transmission Power Control

As shown in Table 2, when the SAFEMITS node's RF transmission power is not controllable, the Pairwise-BD hybrid is most energy-efficient at lower communications ranges, whereas the Pairwise-SKDC hybrid is most energy-efficient at greater communications ranges. However, the Pairwise-BD hybrid benefit only occurs when multicast transmission is available, thus demonstrating the importance of this capability to key management energy efficiency.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)			
	Pairwise Only	Pairwise-GDH Hybrid	Pairwise-BD Hybrid	Pairwise-SKDC Hybrid
30	.67	*.70*	.46	**.45**
35	1.14	*1.15*	**.66**	.71
40	*1.98*	1.80	**.93**	1.05
45	*3.21*	2.78	**1.35**	1.46
50	*4.95*	3.98	1.80	**1.44**
60	*11.80*	8.19	3.38	**1.58**
70	*23.27*	14.84	5.76	**2.77**
80	*42.24*	23.13	8.60	**4.59**
90	*70.80*	28.96	10.65	**7.24**

Table 2 - Hybrid Keying Average Energy Consumption, No Transmit Power Control

Although our research has identified key management energy-efficiency improvements for a number of scenarios, further improvements are possible. We have identified the following areas were additional research would enhance key management performance:

- Development of an optimized group determination algorithm – The algorithm we are currently using is sub-optimal since it simply finds the largest group available, whereas a smaller group may provide a greater reduction in energy consumption depending on the relative positions of the group members. Furthermore, the optimal amount of overlapping between groups has not been determined.

- Finding GDH-optimized groups and optimizing member roles – Finding singly-hop-connected groups is an overly restrictive simplified approach towards performing hybrid keying using a Group Diffie-Hellman protocol. A stricter approach would not require all members to interconnect, but rather simply require the controllers to connect to all members and require all non-controlling nodes be connected to their protocol "next-door

neighbors". Moreover, selection of member roles can be further optimized to minimize the communications energy by having protocol "next-door neighbors" to match with the actual physical "next-door neighbors".

- Multiple group keying protocols in a single hybrid protocol – Thus far, we have examined hybrid protocols that included a group keying protocol and a pairwise keying protocol. We believe there are scenarios, especially with much larger SAFEMITS networks, where two or more different group keying protocols in addition to a pairwise keying protocol may provide a better hybrid protocol than just one group keying protocol alone.

- Multi-hop keying – Although establishing keys via protocols that require multiple hops appears to be less energy efficient, we believe there may be scenarios in densely populated SAFEMITS networks where multi-hop keying may be effective.

- Parasite keying – We have qualitatively identified scenarios where Parasite keying is advantageous, but have not yet shown these benefits quantitatively. These benefits are best shown via simulation in the near-term, building upon our existing hybrid keying protocols.

- Per-node group determination and role determination algorithms – As we transition from research and simulation to integration and demonstration, it is important we appropriately transform our algorithms as well. Currently, our algorithms assume a degree of omnipotence regarding the locations of neighboring SAFEMITS nodes, the possible interconnections, the groups to be formed, and each node's given group role. Our algorithms must assume less to handle the asynchronous nature of self-assembling networks, including making determinations with limited information that may result in sub-optimal configurations.

- Integration of routing and keying protocols – Despite the additional complexity of integrating routing and key establishment protocols, there may be significant advantages in combining some aspects of these protocols. For instance, some key establishment protection is necessary to protect routing determination protocols. However, some multi-hop key establishment protocols require routing to already be determined. Integrating portions of both protocols together may provide energy reductions not possible with these functions separated.

- Further protocol exploration – As we develop increasingly more sophisticated simulations and development demonstrations, new issues with the protocols will become important. Different communication channel models will have varying impact on the latency of different cryptographic protocols and on the ability of the network to run multiple protocols concurrently. The effect of SAFEMITS node dozing on different keying protocols must be examined and deficiencies addressed. Asymmetric communication links between nodes seriously impact the use of certain key management protocols. Further development of amortization techniques and simulating / testing is needed in order to minimize energy consumption and latency.

- Key management during *post routing-establishment* and network re-seeding phases – Once the *routing infrastructure* has been established SAFEMITS nodes can utilize (at a cost) remote resources. During network re-seeding (or during significant network disruptions) the network may consist of regions that for a moment completely lack routing, regions that have well established routing, and mixed regions with only partial, highly irregular routing in place. The chaotic nature (and potentially low latency tolerance) of such situations will be especially challenging to energy efficient key establishment. The joining of two SAFEMITS networks also presents similar challenges to key management. Both of these phases will require different key management protocol mixes than the mixes used during the *pre-routing* and *routing establishment* phases

- Researching other security services – Key management is but one of the many security services that must be supported by the distributed SAFEMITS network. Our research should additionally examine other security services, such as integrity, authentication, and non-repudiation, to determine efficient and secure methods of providing these services.

- Implementation of security services – Finally, we must validate our research via SAFEMITS prototype-based experiments. The challenges of implementation and the many real world issues such as energy, latency, and network self-assembly provide an excellent environment for identifying the critical elements of the research. In addition to the key management, a prototype implementation must include basic security services such as confidentiality, integrity, and authentication. An independent red-team security analysis of our design and implementation will also provide great value to this security research.

1 Introduction

Distributed SAFEMITS networks (DSNs) will produce high-quality battlefield information using large numbers of physical SAFEMITSs (e.g., acoustic, seismic, visual) communicating via ad hoc wireless networking. Advances in microelectromechanical systems (MEMS) technology allow SAFEMITSs to be re-programmable in the battlefield, self-localizing, and to support low-energy, wireless, multi-hop networking, while requiring only minimal pre-configuration. To reliably support coordinated control, management, and reporting functions, SAFEMITS networks will be self-organizing with both decentralized control and autonomous SAFEMITS behavior, resulting in a sophisticated processing capability.

Battlefield constraints create daunting engineering challenges for SAFEMITS designers. SAFEMITS packages will be small, lightweight, inexpensive, and low-power. Distributed in irregular patterns across remote and often hostile environments, SAFEMITS nodes will autonomously aggregate into collaborative, peer-to-peer networks. SAFEMITS networks must be robust and survivable despite individual node failures and intermittent connectivity. Support for lengthy mission lifetimes constrains battery consumption to miserly rates when not in an energy conserving dormancy. High information assurance must be provided despite the use of unattended SAFEMITS packages with relatively weak resistance to tampering.

Distributed SAFEMITS networks will be a mission critical component requiring commensurate communications security protection. Warfighters must be assured that received SAFEMITS information is correct. Deployed SAFEMITSs must only accept legitimate queries, commands, and software updates. SAFEMITS network communications must prevent disclosure and undetected modification of exchanged messages.

Providing confidentiality and authentication is critical to preventing an adversary from compromising the security of a distributed SAFEMITS network. However, providing key management for confidentiality and group-level authentication is difficult due to the ad hoc nature, intermittent connectivity, and resource limitations of the distributed SAFEMITS network environment. This DARPA-sponsored research will address this problem by identifying and developing cryptographic protocols and mechanisms that efficiently provide key management security support services.

Operating in a hostile environment exposes unattended SAFEMITSs to a myriad of threats that require various forms of physical, communications, and cryptographic protection. Our research focuses on only one security problem in this security space: key management for confidentiality and group-level authentication in resource-limited distributed SAFEMITS networks. This research addresses the problems of achieving sufficient "trust" among unattended SAFEMITS nodes to support key management, and efficiently performing cryptographic key computations for message privacy and authentication. This research does not address physical protection of SAFEMITS node processing, efficient algorithms for performing data confidentiality and message authentication, or key management for other SAFEMITS node functions such as frequency hopping and spread spectrum communications and Global Positioning System (GPS) keying.

SUNYIT research team is collaborating with other research programs examining SAFEMITS and SAFEMITS network technology. In addition to DARPA's SAFEMITS program, the Army Research Laboratory's (ARL) Advanced Telecommunications and Information Distribution Research Program (ATIRP) consortium and the Internet Engineering Task Force (IETF) Mobile Ad-Hoc Networking (MANET) working group are exploring SAFEMITS network solutions. For ARL's ATIRP consortium, SUNYIT research team is developing a communications security architecture that protects Army SAFEMITS networks under battlefield constraints. The ARL-sponsored architecture is broad, examining a wide range of threats, attacks, vulnerabilities, requirements, constraints, and

corresponding security services. Conversely, our SAFEMITS research is narrowly focused, examining the key management security support service in great depth.

This document describes our SAFEMITS network constraints and key management approaches research for FY 2000. The remainder of this document is organized as follows:

- Section 1 provides background on distributed SAFEMITS networks.
- Section 3 describes the security requirements applicable to the problem of establishing keys for confidentiality and group-level authentication.
- Section 4 describes constraints that make this problem difficult.
- Section 5 describes and analyzes protocols that can be used to establish keys between group of SAFEMITS nodes.
- Section 6 examines network-wide approaches to optimizing energy consumption for various SAFEMITS network scenarios.
- Section 7 describes additional areas of research that we have identified that could further enhance SAFEMITS network key management performance.

2 Background

2.1 SAFEMITS Node Technology

The SAFEMITS node is the basic component of the SAFEMITS network. Nodes are designed for ease of deployment and to be low cost, compact, lightweight, and disposable. Local and collaborative signal processing across the wireless network enhances SAFEMITS nodes primitive communications functions (e.g., seismic, acoustic, magnetic). The following sections describe features of these nodes that support the basic functions of a SAFEMITS node, including: event detection, event or target classification, target tracking, and event reporting.

2.1.1 SAFEMITS Node Hardware

SAFEMITS nodes provide the core communications functions of the SAFEMITS network. The SAFEMITS node hardware design and communications architecture are greatly influenced by their finite battery limitations. SAFEMITS node designs by SAFEMITSia Corporation [WINSNG00], Rockwell Collins [Agre99, Agre00b], and the Army Research Laboratory (ARL) [Falco00] reflect this design constraint through their use of low-power hardware and embedded processors. This section describes capabilities and design characteristics of these SAFEMITS nodes.

2.1.1.1 Hardware Design

In order to simplify deployment and support ad hoc network formation, we assume that future SAFEMITS nodes will support a flexible hardware and software architecture allowing them to take on various roles in the network (e.g., gateway vs. communications node) and support various SAFEMITS applications. The exact function of each SAFEMITS node may not be determined until deployment and may change over the course of its mission. Flexibility is an important requirement in reducing the amount of equipment needed by soldiers in the field in order to deploy a SAFEMITS network and in supporting remote deployment techniques (e.g., airdrop). In general, we assume that SAFEMITS nodes support the following functions or features [Mills00, Tassiulas00a, Tassiulas00b]:

- Dynamically configurable to support a variety of network functions or roles (e.g., gateway, ordinary node);
- Remotely re-programmable to support new functionality (e.g., new signal processing algorithms);
- Support location determination mechanisms to define their exact or relative position (e.g., the Global Positioning System (GPS) or localizing functions such as the radio frequency Localizer by AEther Wire Location, Inc. [Aether95]);
- Support low-energy networking to exchange data locally over a wireless multi-hop ad hoc network;
- Support long-haul communications capabilities for exchanging data over long-haul radio circuits (i.e., when designated as a gateway node); and
- Require only a minimal pre-configuration prior to deployment.

Energy is the most constraining factor in SAFEMITS node design affecting all aspects of a SAFEMITS node design. Microprocessor selection is one area where energy conservation is important. There are numerous commercially available microprocessors designed for embedded low-power environments. These microprocessors are suitable for both commercial applications (e.g.,

cellular phones, PDAs) and SAFEMITS nodes with similar energy constraints. Current embedded processors typically have power dissipations less than 500 mW (see Figure 2), operate with clock speeds of less than 200 MHz, and require voltage supplies in the range from 1.0 to 3.3 volts. Energy constraints of embedded processors will be discussed more in Section 2.1.1.2 and Section 4.1.1.

Since 1965, the prediction made by Gordon Moore from Intel that the microchip transistor density will double every 18 months has proven remarkably true [Wittman99]. According to Wang [Wang00b], the Semiconductor Industry Association forecasts that to keep up with Moore's Law, portable-computing platforms will reduce voltages from today's typical 1.5 V to 0.3 V in the year 2014. This will allow processing to improve while maintaining low-power consumption. Current research in the communications circuits designed for SAFEMITS nodes have produced designs that consume power in the nanowatt range [Sarneke98].

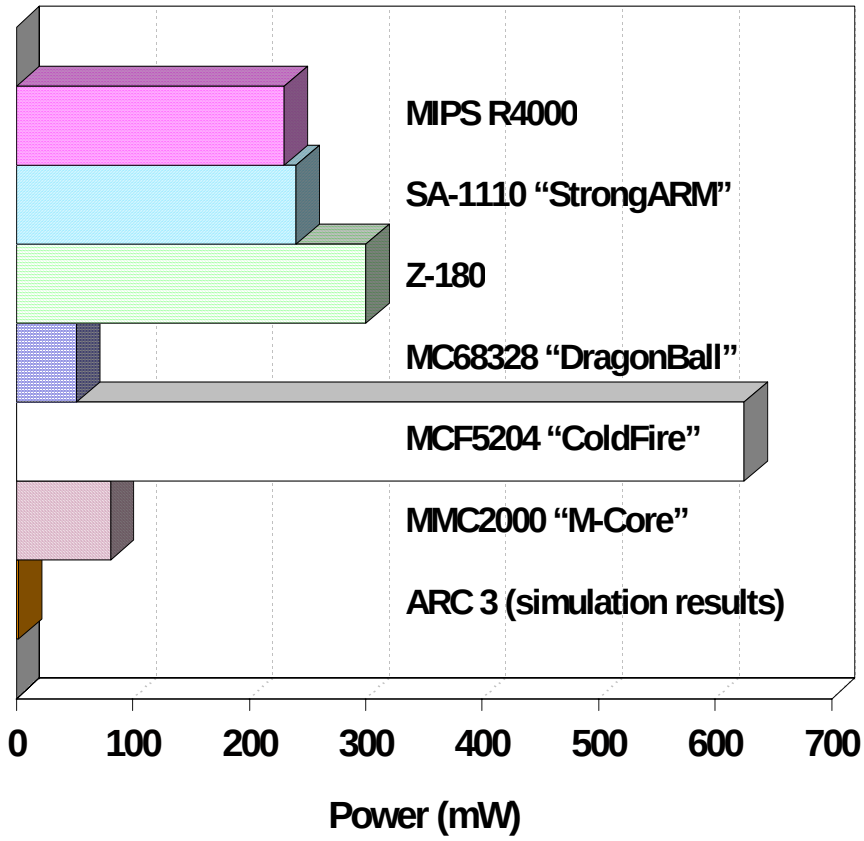

Figure 2 - Power Dissipation for Selected Embedded Processors

There are several SAFEMITS nodes that have been recently demonstrated in SAFEMITS mission environments. Both the Rockwell Science Center[1] and SAFEMITSia Corporation[2] have a Wireless Integrated Network SAFEMITS node suitable for low-energy SAFEMITS networks. The SAFEMITSia node uses a low-power pre-processor, the Zilog Z-180, and a low-power core processor, MIPS R4400, for performing signal processing functions. The Rockwell node uses a low-

[1] http://wins.rsc.rockwell.com/

[2] http://www.SAFEMITSweb.com/

power Intel StrongARM SA-1110 microprocessor. Both the MIPS at 80 MHz and StrongARM at 133 MHz typically consume less than 300 mW when in the *run* mode. The StrongARM consumes less than 1 mW in its sleep mode.

2.1.1.2 SAFEMITS Node Energy

Perhaps the greatest limiting factor in a SAFEMITS node's life expectancy is its battery capacity. In general, we assume that SAFEMITS nodes have a limited battery capacity and therefore must take precaution to conserve their energy. Energy conservation is applicable at the node level and at the network level. For example, the routing decisions made by one node or a group of nodes can affect the energy levels nodes receiving their traffic. Some energy conscious routing algorithms attempt to balance available energy throughout the network [Mills00, Tassiulas00a, Tassiulas00b]. We assume that once deployed, the SAFEMITS node battery cannot be recharged or replaced while in the field. However, at least one SAFEMITS node developer is investigating this possibility using solar arrays to add recharging capabilities.[3] Nonetheless, this application may not be suitable for all types of SAFEMITS network missions (e.g., remote reconnaissance).

Energy is the capacity to perform work and is related to the power consumed over time. Within a microprocessor, power consumption is related to the frequency of the supplied clock and the voltage supply. The approximate power consumed by an integrated circuit is a function of voltage (V), clock frequency (f), capacitance (C), and quiescent current [Kurkowski00]:

$$P = V^2 \cdot f \cdot C + P_{static}$$

P_{static} is associated with the semiconductor's physical characteristics, temperature, and its voltage and current supply. The remaining part of the equation is dynamic and can be controlled by changes to the chips voltage supply and clock frequency. A reduction of clock frequency alone will not reduce energy because the frequency is inversely proportional to time. Therefore, a reduction in clock frequency without a reduction in voltage will provide no benefit. The software will simply take longer to run and thus offset the power savings of having a slower clock. A reduction in voltage may be possible when lowering the clock frequency because a slower clock may allow longer gate settling times resulting from the lower voltage [Lorch95, Lorch98]. However, a slower clock may mean that other components are powered for longer durations (e.g., RAM), negating any power savings of a slower processor clock.

The actual amount of energy available by a SAFEMITS node's battery is a function of temperature, the rate of dissipation, and the battery technology. A battery's potential capacity is measured in milliampere-hours (mAh) and is a function of the ambient temperature and the load on the battery. Capacity can also be represented as energy (Joules) that is the amount of power consumed over time. Table 3 shows some typical battery capacities.

[3] Conversation with Jon Agre from Rockwell Science Center on 17 March 2000.

Battery Model #	Typical Capacity, Nominal Voltage	Chemical Composition	Energy Potential[4]
CP8136Energizer	1200 mAh @ 3.6 V	Ni-MH rechargeable	12.96 kJ @ 3.0 V
MN1500Duracell AA	2850 mAh @ 1.5 V	Alkaline	15.39 kJ @ 1.5 V
MN2400Duracell AAA	1150 mAh @ 1.5 V	Alkaline	6.21 kJ @ 1.5 V

Table 3 - Battery Charge and Power

As an example of battery energy capacities for possible SAFEMITS nodes, we reference the capacity available to the WINS NG by SAFEMITSia. The SAFEMITS node's 7.2 volt battery pack supplies roughly 26 kJ of energy[5]. At a data rate of 10 kbps transmitting with an RF power of 10 mW and a radio subsystem dissipation of 210 mW, the transmission energy rate is 21 μJ/bit and will communicate approximately 900 meters[6]. Shorter distances require less energy. Similarly, the receive energy rate is 14 μJ/bit for a radio subsystem dissipation of 140 mW.

In order to conserve energy, embedded processors typically have low-power modes that slow or halt the processor clock and place the device in a state that consumes less power. Newman and Hong [Newman98] examined the power consumption of the Palm III in various modes with its processor, the Motorola DragonBall (e.g., sleep, doze, run). In *sleep* mode the unit appears off, with many peripherals in an energy-conserving low power mode. An interrupt from a physical button or the real-time clock will wake the system. In its *doze* mode, the processor is halted but some peripherals including its display are powered. The recovery from the doze mode is faster than the sleep mode. In the *run* mode, the device is fully powered and the processor is executing instructions. A true energy savings for the Palm Pilot can be gained by returning the processor to the sleep or doze mode that uses less power [Newman98]. In a bursty communications or processing environment, faster and more efficient software allows the device to return to this state faster. The energy saved by reducing the scalable clock frequency of the device is negligible without a corresponding voltage reduction and in some cases is offset by unreliable performance caused by a slower clock frequency.

In order to conserve power, we assume the majority of a SAFEMITS's lifetime is spent in a low power *doze* or *sleep* mode. In some cases, the SAFEMITS node may only be placed in a run mode upon event detection or to route traffic from a neighboring node. The time in the power consuming run mode should be minimized to conserve energy. An energy reduction or loss by an individual SAFEMITS node will affect the energy balance in the SAFEMITS network requiring routing tables to be resynchronized to avoid the weak SAFEMITS node and avoid potential isolation problems in the network.

2.1.1.3 SAFEMITS Node Mobility

We assume unattended ground SAFEMITSs have no mobility. Once deployed, the SAFEMITS nodes will not be physically moved. However, in some mission environments, SAFEMITS nodes may be replaced if the battery supplies are low or the nodes are damaged. For remote reconnaissance missions, hand replacement is not possible. Instead, nodes are added to the network using random

---

[4] Actual available energy is a function of voltage, temperature, and discharge rate. The alkaline potential is based on 21° C (i.e., operating range –20° C to 54° C). Ni-MH based on a C/5 discharge to 0.9 volts per cell.

[5] Provided by William Kaiser.

[6] For a 1-kilobit data payload over a BPSK surface-to-surface link with a Free Space Rayleigh channel propagation law of $1/R^4$ and with an error rate of 10^{-6}.

"seeding" methods. In both cases, the makeup of the network changes over time as nodes are added or deleted from the network.

2.1.1.4 Communications Capabilities

We assume that SAFEMITSs may contain any number of capabilities including seismic, acoustic, magnetic, infrared, radar, and video. Although the communications of events is done in real-time, the reporting of events may not. Reports may be fused with reports from other SAFEMITS nodes. SAFEMITS nodes may deliver video in real time and require suitable Quality of Service (QoS) to support its delivery.

2.1.1.5 Tamper Detection and Protection

Tamper detection refers to technologies that actively or passively detect tampering of the physical device by an adversary. There are several physical layers to a tamper attack starting first with removal of the SAFEMITS node's cover. Once the internal hardware is exposed, an attacker may alter a node's hardware or software or attempt to extract SAFEMITSive information from the SAFEMITS node memory. After power is removed from a memory device (i.e., RAM), remnants of past data may still exist [Anderson97]. In the SAFEMITS network environment, this may pose a security risk to SAFEMITS nodes when their batteries become exhausted. Without battery power, a SAFEMITS node may not be able to actively provide tamper protection.

Tamper protection may consist of both passive and active components and be applied to the physical or electrical design. Active technologies detect the act of tampering and perform some countermeasure such as zeroizing memory. Passive technologies deter or delay an adversary from extracting SAFEMITSive data from the SAFEMITS node. Because of the low cost and disposable design of SAFEMITS nodes, we assume that their tamper capabilities will be limited.

2.1.2 Software

We assume that SAFEMITS nodes will employ an embedded operating system to manage and support its applications for providing real-time performance. Although the SAFEMITS applications running on the node may be custom, the underlying operating system may be an embedded commercial-off-the-shelf (COTS) operating system. For example, the SAFEMITSia WINS NG uses the Microsoft Windows CE operating system found in commercial PDAs. We assume that the operating system will be trustworthy but not a "trusted" operating system as defined by the National Computer Security Center (NCSC). As defined in *Department of Defense Trusted Computer System Evaluation Criteria* [DoD85], we assume the operating system will have the lowest possible rating of *Class D Minimal Protection*.

Additionally, in order to support the implementation of any security requirements, we assume that the embedded operating system is not bypassable, and properly implements the documented interfaces. Furthermore, the implementation must provide assurance that it does not allow any unintended execution paths or access. We do not assume any specific security functionality from the operating system. In order to assure that the operating is properly implemented, evaluation methodologies such as the *Common Criteria for Information Security Evaluation* [Common99] may be employed.

In support of a flexible design, we assume that SAFEMITS nodes support remote reconfiguration and reprogramming to incorporate flexibility into their design. We also assume that SAFEMITS nodes may support the use of mobile software. A possible use of mobile software to perform vulnerability assessments is described in [Barrett98, Dumas99].

2.2 SAFEMITS Network Missions

The primary mission of the SAFEMITS network is to detect and report events occurring within range of the SAFEMITS network. The SAFEMITS nodes in the network generally have crude communications functions (e.g., seismic, magnetic). Through the cooperation of other nodes in the SAFEMITS network, a more reliable communications function is possible. Using their communications capabilities, individual SAFEMITS nodes can detect events such as movement of dismounted troops, movement of armored vehicles, detection of chemical occurrences, etc. Once an event is detected, the detecting SAFEMITS node may report the event directly to a remote command and control (C^2) application or collaborate with other SAFEMITS in the network to more reliably identify and track a target. Some basic missions of SAFEMITS networks include border monitoring or perimeter defense, and remote reconnaissance or surveillance.

In these mission scenarios, we assume that detected events (e.g., troop movement) will be reported to a C^2 application. The location of the application can be local (e.g., a dismounted soldier in the SAFEMITS field) or remote (e.g., a centralized command and control center). Reports may be delivered immediately upon event detection or cached for later delivery. Delivery may be upheld to prevent detection (i.e., LPD) or to fuse with other reports from neighboring SAFEMITSs.

2.2.1 Perimeter Defense or Area Denial

SAFEMITS nodes can be deployed in a network to detect and report the movement of targets across a defended border or perimeter. In this scenario, SAFEMITSs may be hand placed in a one-dimensional fashion (e.g., along a fence line) to detect and report perimeter violations. The network will typically be static with few additions or deletions over its lifespan. Although the SAFEMITS may be located in close proximity to friendly troops, the nodes may be unattended for long periods of time.

In a Military Operations on Urbanized Terrain (MOUT) operation, dismounted soldiers place SAFEMITSs within sections of a building as those sections are "cleared" similar to a perimeter defense mission. The network topology will change as the perimeter advances and more nodes are added to the network.

2.2.2 Remote Surveillance

In remote surveillance missions, SAFEMITS nodes may be deployed in remote areas behind enemy lines. A two-dimensional SAFEMITS field is formed with randomly placed nodes. Mills [Mills00] assumes a typical placement of 100 meters (actual deployment techniques have not been developed). The collection of SAFEMITS nodes within the SAFEMITS field can be used to remotely monitor targets passing through the SAFEMITS field. We assume that the SAFEMITSs can be programmed to report back to a C^2 application either when an event happens or cache or fuse reports for later delivery. In addition, the SAFEMITSs will accept commands from the C^2 application (e.g., sleep, change report interval). The returned reports could be used for intelligence gathering or to direct assets against a detected enemy. The SAFEMITSs will be unattended for long periods of time, possibly until their batteries are depleted.

2.3 SAFEMITS Network Architecture

This section discusses details of the SAFEMITS network architecture including its environment (Section 2.3.1), the types of data exchanged (Section 2.3.2), and its communications architecture (Section 2.3.3).

2.3.1 Environment

The SAFEMITS network environment can be physically demanding on SAFEMITS nodes. The nature of the unattended SAFEMITS network missions place the nodes in areas where they are subject to physical destruction, theft, and exposed to extreme temperature conditions. The communications environment may also be difficult due to interference and fading due to ground placement and foliage [Wang00a].

The population density of SAFEMITS nodes in the network may vary depending upon the application (e.g., boundary defense, surveillance), the communications capabilities of the SAFEMITS nodes, and the environment (e.g., desert, rain forest). For example, a one-dimensional boundary application may require SAFEMITS nodes be placed every 100 meters in a line. The same application may require a closer placement in a denser terrain that limits signal propagation. A two-dimensional surveillance application requires a different population density. In general, we assume relatively short spacing to provide low-probability of interception (LPI). In all scenarios, we assume that once deployed SAFEMITS nodes have no mobility. This implies that the network is somewhat static. However, although nodes are not mobile, the topology of the network may change as nodes are added or deleted from the network. Nodes may be added to replace nodes that have lost power or were destroyed.

2.3.2 Data Types

Within the SAFEMITS network, the amount and type of data exchanged is greatly influenced by the battery and energy constraints of the SAFEMITS nodes. As noted by Kaiser and Pottie [Kaiser00], the energy required to transmit a bit can be much greater than the cost to internally process a bit. For this reason, raw data will typically be processed locally and the results exchanged within the network with fewer transmitted bits and less energy consumed. The data exchanged within the SAFEMITS network may include raw SAFEMITS data, SAFEMITS node event reports destined for a remote command center or dismounted soldier in the field, or SAFEMITS commands and controls.

In order to conserve energy, raw SAFEMITS data is not usually forwarded within the network but processed locally into event reports that may include target classification and direction information. As reports propagate through the SAFEMITS network, intermediary nodes may fuse their reports with those from other neighbor nodes to assist in the classification of targets and the tracking of targets. *Data fusion* helps to reduce the total amount of bits of data routed within the network. Nodes in gateway locations within the network may perform data fusion functions for other local SAFEMITS nodes. However, we assume for some SAFEMITS applications raw real-time data such as voice or video may be exchanged without significant local processing.

SAFEMITS reports are eventually forwarded to a C^2 application. The C^2 application may be remote (i.e., accessible via long-haul communications links) or local (e.g., a dismounted soldier in the field). Commands issued from the C^2 application may include request for communications reports or commands requesting the SAFEMITS perform some function or revert to a certain processing state (e.g., asleep, awake). Commands may target a particular SAFEMITS node or a group of SAFEMITS nodes. We assume that groups may be defined based on physical location, by SAFEMITS function, or some other SAFEMITS node specific criteria (e.g., by a cluster group as defined in [Wang00a]).

2.3.3 Communications Architecture

The distributed SAFEMITS network is an ad hoc wireless network where the membership and roles of SAFEMITS nodes is generally not known until the deployment of the network. SAFEMITS nodes may be deleted permanently from the network when their available energy falls below acceptable limits or temporarily when they return to a sleep state. Once deployed, the network is

self-organizing, developing a routing topology that provides strong connectivity throughout the network (i.e., a path exists between every node) [Mills00, Tassiulas00a, Tassiulas00b]. This process will inherently remove isolated nodes from the network. In order to maintain the energy balance within the network, re-organization is required throughout the life of the network as nodes are deleted and added to the network. This creates a fault tolerant network design where the loss of a fraction of the nodes causes a graceful degradation in network performance.

We assume that the SAFEMITS network will support a layered protocol stack similar to that shown in Figure 3. The physical layer provides a wireless link between neighboring SAFEMITS nodes (see Section 2.3.3.1) while the network layer allow for routing and delivery of data throughout the network (see Section 2.3.3.2). We assume that some applications may require reliable delivery services from a transport layer (see Section 2.3.3.3). Various SAFEMITS applications are supported at the application layer (see Section 2.3.3.4).

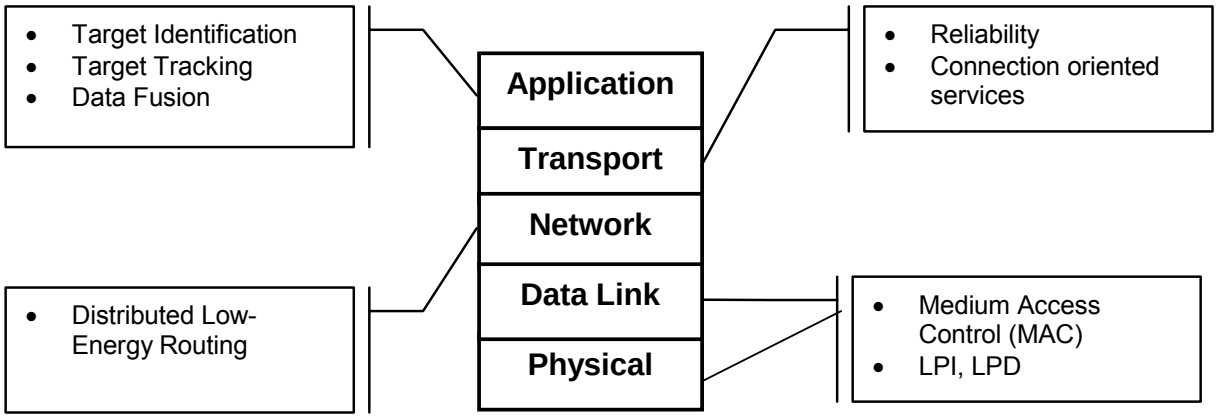

Figure 3 - SAFEMITS Network Communications Layers

2.3.3.1 Physical Layer

The physical layer defines the mechanisms for medium access control (MAC) for the wireless SAFEMITS network. There are various physical layer MAC protocols that may be used for SAFEMITS routing. In general, for distributed SAFEMITS networks, distributed control is preferred over centralized control for survivability reasons (e.g., base-station) [Tayong00].

We assume that nodes have variable control over their radiant RF energy allowing them to dynamically control the range of their communications and provide a lower probability of interception (LPI). Typically, this range is less than 100 meters, with gateway nodes having additional communications capabilities and power to support long haul communications to reach a remote command center (e.g., more than 1 mile).

We assume that the SAFEMITS network is a *low-bandwidth* network with a typical packet size of 30 bytes and a transmission rate of no more than 1 packet per second [Mills00]. Both the Rockwell and SAFEMITSia[7] nodes support data rates over 10 kbps with the Rockwell node supporting a rate of 100 kbps [Agre99].

[7] Provided by William Kaiser.

2.3.3.2 Network Routing Layer

The SAFEMITS network utilizes a multi-hop bursty packet based network routing protocol to deliver data throughout the network. The finite energy of the network is the primary design constraint in developing a low-energy routing algorithm that balances energy throughout the network. Over time, nodes that are a focal point for network traffic will lose energy more quickly than those nodes at the edges of the network. For this reason, we assume that routing protocols like those clustering protocols described by Mills [Mills00] and Wang, et. al. [Wang00] will periodically re-organize themselves to balance energy dissipations in the network and extend the overall life of the network

Self-organization is required at time of deployment to initialize routing tables without the assistance of a human administrator. Later, re-organization is also necessary because of the ad hoc nature of the network – SAFEMITS nodes may be added and deleted from the network over its lifetime. For example, in a MOUT application, nodes may be added as the defensive perimeter expands. Nodes may be deleted as their energy levels fall below acceptable levels or if they are physically destroyed. In order to maintain strong network connectivity in these conditions, periodic re-organization is necessary. The re-organization will change the topology of the network, possibly changing the neighboring nodes that were known and trusted by a SAFEMITS node.

As a multi-hop network, packets are transferred from node to node until they reach their final destination. Intermediary nodes make routing decisions based on their routing tables that are constructed based on link costs that consider the energy to transmit and receive. Intermediary access requires visibility to the packet header field, implying that portions of the header may not be encrypted at the network layer. Intermediate access also makes it possible to perform data fusion at the network layer in order to reduce the number of bits transmitted to the next link. For example, a node could combine data packets or delete redundant information (e.g., commands) sent to a group of nodes.

In order to construct the necessary routing information, each SAFEMITS node must determine its neighbor nodes and then make a determination of which it will route traffic to. The decision on how to do this energy efficiently is dependent on the routing algorithm. Cluster routing algorithms like those described by Mills [Mills00] and Wang, et. al. [Wang00a] discuss a concept of clusters of nodes centered on a cluster head (see Figure 4). Typically, a cluster will consist of fewer than 10 nodes with the entire network consisting of fewer than 1000 nodes randomly spaced roughly 100 meters apart.[8] Local nodes communicate with the network through their cluster heads. Nodes lying between clusters serve as gateways between the clusters. Some gateways may also support long-haul communications to a remote C^2 application.

[8] Conversation with Dr. Diane Mills of Lockheed Sanders on 17 February 2000.

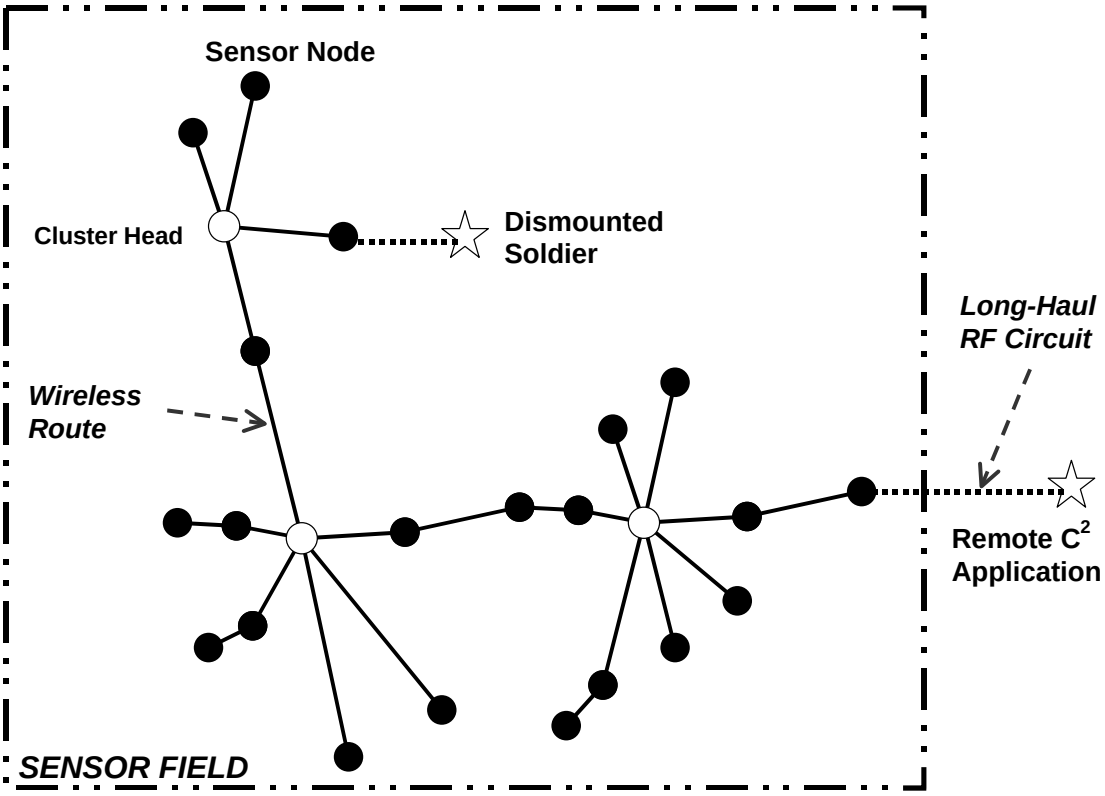

Figure 4 - Cluster Algorithm-based Routing

2.3.3.3 Transport Layer

The transport layer protocols can provide reliability and session control for SAFEMITS node applications. The majority of SAFEMITS network communications are bursty, packet-oriented communications that do not require the reliability of the transport layer. We generally assume that all SAFEMITS node communications are unreliable.

Using real-time multi-media applications over the SAFEMITS network may require the ordering and reliability mechanisms of transport layer protocols. Real-time communications may include the relaying of audio or video back to a C² application from a SAFEMITS node. For this type of data, resource reservation functions are important in maintaining a level of service between the sender and receiver [Ephremides00].

2.3.3.4 Application Layer and Data Fusion

As we noted earlier in Section 2.1.1.2, depending on the SAFEMITS node hardware, it is generally more efficient to perform local processing on SAFEMITS data rather than transmit the raw data to a centralized point for processing. SAFEMITS nodes will contain signal-processing algorithms specific to their functionality (e.g., acoustic, seismic) to perform local target identification and perform collaborative target tracking. Tracking information received from a group of SAFEMITSs may be processed by an intermediate fusing node (see Figure 5). The intermediate node may send the resulting report through the network for additional fusing or delivery to a C² application.

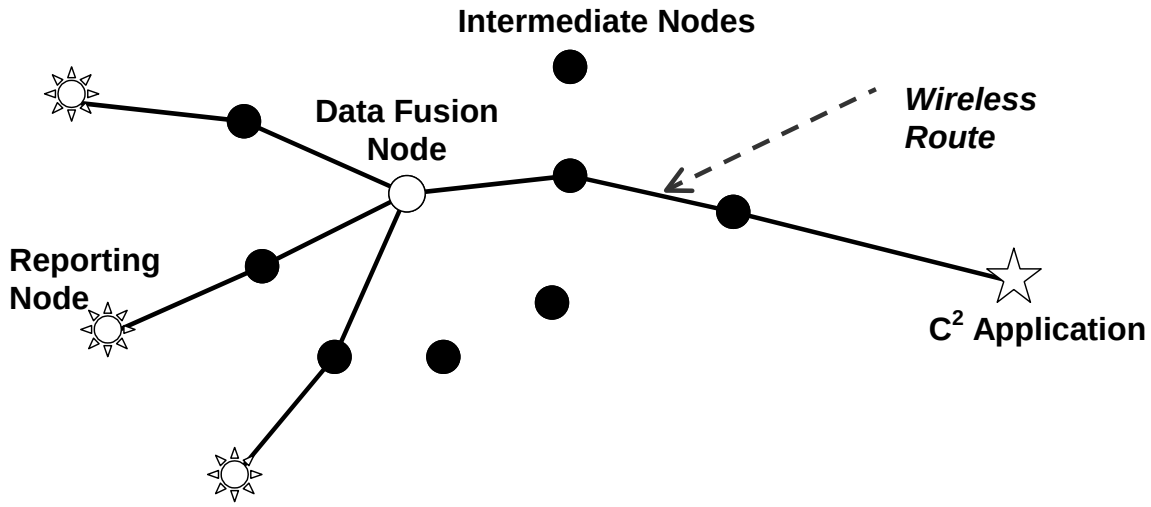

Figure 5 - Data Fusion within a SAFEMITS Network

2.4 Concept of Operations

During its lifespan, a SAFEMITS node may go through a series of stages starting with its manufacture and eventually leading to its deployment in a SAFEMITS network. The following sections describe a generic concept of operations that may be applied to SAFEMITSs nodes and SAFEMITS networks. We assume that a typical operational scenario may include the following steps (see Figure 6), each described in the following sections: manufacture of SAFEMITS nodes, temporary storage of SAFEMITSs at a depot, initialization of SAFEMITSs for deployment, deployment of SAFEMITS nodes, mission operations, and mission completion.

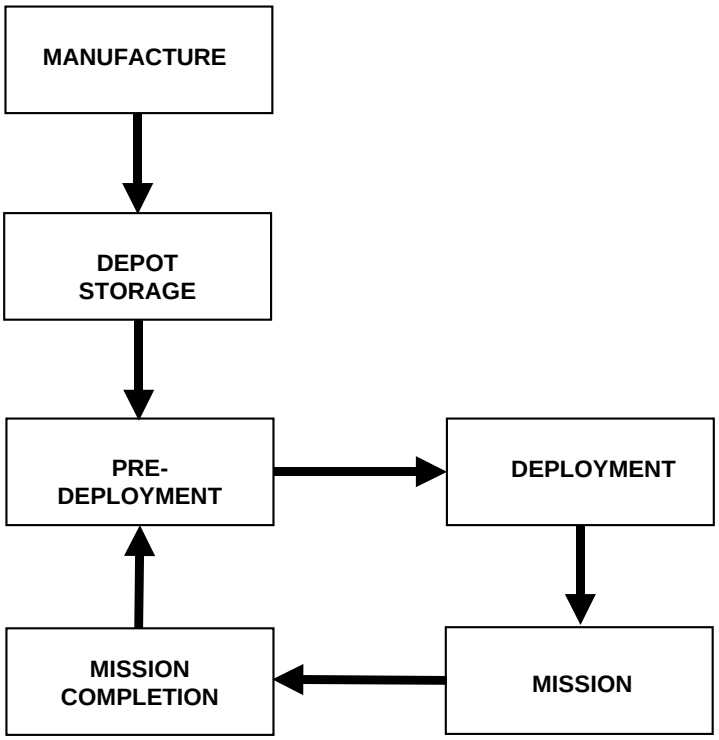

Figure 6 - Sample Concept of Operations for SAFEMITS Networking

2.4.1 Manufacture

During the manufacturing process, SAFEMITS node hardware is assembled and core software is loaded (e.g., operating system, communications drivers). We assume that the SAFEMITS node architecture is flexible, allowing for the addition of various SAFEMITSs and supporting software (e.g., target classification algorithms) during the pre-deployment stage. However, some basic functions may be loaded during manufacture. Other initialization information may also be loaded during manufacture including cryptographic algorithms and key material.

The following assumptions about the manufacturing process may affect the security of the SAFEMITS nodes and network:

- SAFEMITSs will be manufactured in large quantities at low cost;
- Access control to the manufactured SAFEMITSs and SAFEMITS parts may *not* be tightly controlled.
- SAFEMITSs nodes may be susceptible to theft during manufacture;
- The development process may not have tight access control mechanisms allowing unauthorized hardware or software modifications; and
- Software bugs may be introduced due to human error during the development process. These bugs could put the SAFEMITS node in an undetermined state making it susceptible to compromise.
- Portions of the manufacturing process may occur outside the United States.

Following their manufacture, SAFEMITS nodes may be forwarded to a depot prior to distribution to the field. Due to the relatively large quantities of SAFEMITS nodes that may be manufactured, we assume that the manufacturing process will not be tightly controlled. Army depot facilities like Tobyhanna Army Depot have secure facilities for the manufacture and refurbishing of COMSEC equipment. However, this additional security adds to the cost of each SAFEMITS and is contrary to the goal of inexpensive and disposable. We assume that the low-cost disposable nature of the technology discourages use of relatively high-cost secure manufacturing facilities.

2.4.2 Depot Storage

Following manufacturing process, the SAFEMITS nodes may be stored in a depot for extended periods of time awaiting deployment. The depot may also serve as a point for repairing damaged nodes or refurbishing outdated nodes. During this time, we assume that access to the SAFEMITS nodes is not tightly controlled. A lack of access control may make nodes susceptible to theft and allow unauthorized modifications of hardware or software (i.e., tampering).

2.4.3 Pre-Deployment

In order for SAFEMITS nodes to be deployed within a SAFEMITS network, it may be necessary to initialize or pre-configure the nodes. We note that it is a goal to limit the amount of pre-configuration in order to facilitate deployment. However, we believe some amount of pre-configuration will always be necessary to distinguish legitimate SAFEMITS nodes. Depending on the mission, the pre-deployment stage may take place at the depot or in the field by the deploying soldiers. Pre-configuration may include the following changes to the SAFEMITS node:

- Assigning of communications roles and capabilities (e.g., acoustic, seismic);
- Assigning of network roles (e.g., gateway vs. normal node functions);
- Loading of software (e.g., target analysis algorithms); and
- Loading of cryptographic initialization information.

Depending on the security mechanisms employed by a SAFEMITS node, the loading of cryptographic information during pre-deployment may increase the classification of the node. Depending on the classification, the node may require additional physical protection while in storage. Various mechanisms may be used to reduce the classification and thus the need for physical protection. Cryptographic techniques such as key splitting or key sharing can reduce the classification of the key material. Tamper protection and detection mechanisms may also be used (e.g., tamper seals).

2.4.4 Deployment

Once the SAFEMITSs are physically deployed, the SAFEMITS network will attempt to fulfill its communications mission by exchanging data between SAFEMITS nodes, command centers, and other systems.

2.4.4.1 Self-Organization

In order to support the requirements for random and remote placement of SAFEMITS nodes, SAFEMITSs must be able to self-organize themselves without outside assistance. Each SAFEMITS must be able to identify neighbors, and identify efficient routes to other SAFEMITS nodes and/or a gateway.

2.4.4.2 Re-Organization

Because of the ad hoc nature of SAFEMITS networks, the SAFEMITS network topology may change over the SAFEMITS mission lifetime due to the addition, replacement, or deletion of SAFEMITS nodes. Therefore, it may be necessary to change the routing configuration of the network in order to maintain an energy efficient system in order to maintain a balance of energy throughout the network.

2.4.5 Mission Completion

A mission may complete either because the reason for deploying the SAFEMITSs no longer exists or because the SAFEMITS nodes have died. SAFEMITS node death can be the result of physical destruction, isolation, or depletion of battery energy. In any case, SAFEMITS nodes left behind on the battlefield could be refurbished and/or modified by an adversary and used to attack other operational SAFEMITS networks.

2.5 Environment

The SAFEMITS network environment can be physically demanding on SAFEMITS nodes. The nature of the unattended SAFEMITS network missions place the nodes in areas where they are subject to physical destruction, theft, and exposed to extreme temperature conditions. The communications environment may also be difficult due to interference and fading due to ground placement and foliage [Wang00a].

The population density of SAFEMITS nodes in the network may vary depending upon the application (e.g., boundary defense, surveillance), communications capabilities of the SAFEMITS nodes, and environment (e.g., desert, rain forest). For example, a one-dimensional boundary application may require SAFEMITS nodes be placed every 100 meters to form a line. The same application may require a closer placement in a denser terrain that limits signal propagation. A two-dimensional surveillance application requires a different population density. In general, we assume relatively short spacing, inherently facilitating low-probability of interception (LPI) due to use of low-power communications. In all scenarios, we assume that once deployed SAFEMITS nodes have no mobility. This implies that the network is somewhat static. However, although nodes are not mobile, the topology of the network may change as nodes are added or deleted from the network. Nodes may be added to replace nodes that have lost power or were destroyed.

3 Requirements

The key establishment protocols and approaches for distributed SAFEMITS networks must satisfy several security and functional requirements. The keying protocol must establish a *shared* key (or keys) that can be used by two or more SAFEMITS nodes to provide confidentiality and group-level authentication of application data. The protocol must establish a key between all SAFEMITS nodes that must exchange application data securely, which usually means establishing keys between all one-hop neighbors within the SAFEMITS network. A single key may protect data over a large portion of the network, or just a pair of nodes, with commensurate security ramifications. The following sections detail additional key management requirements.

3.1 Confidentiality

The shared key established by the key management protocol, and its contributing key material, must be protected from disclosure to authorized parties. Similarly, public SAFEMITS information, such as SAFEMITS identities and public keys, should also be encrypted to protect against traffic analysis. Confidentiality should be provided by keys with as small a scope as possible (i.e. fine key granularity) to discourage a single break from compromising a large portion of the SAFEMITS network. That is, establishing unique keys between every pair of communicating SAFEMITS nodes is preferable, in a security sense, to using a single network-wide key.

3.2 Authenticity

At a minimum the access to the *shared* key should be limited to only those parties identified in the protocol, (e.g. implicit key authentication, or data origin authentication of the shared key). Stronger levels of authenticity (e.g. explicit key authentication) are provided by some key establishment protocols. However, most DSN scenarios do not require the extra "assurance", and can verify key delivery by using a system / application protocol.

3.3 Integrity

The *shared* key must not be modified by, its probability distribution (i.e., range of possible values) influenced by, or otherwise a function of the actions of outsiders of the protocol. In other words, an adversary should not be able influence the value of the shared key.

3.4 Freshness

A key establishment process ideally should guarantee its participants that each shared key (session key) is fresh (i.e. has not been reused by one of the participants). This guarantee should include a guarantee that a key used in one cryptographic association has not been used in another association.

Key establishment provides one of two forms of freshness guarantee. The weaker form is provided by key transport where one or more of the participant must depend on some other participant to correctly follow the protocol for the shared key to be fresh. The stronger form, key agreement, allows each correctly operating participant to prove to itself that the shared key is fresh. Typically shared keys need to be changed over time (i.e. rekeyed) for a number of reasons:

- **Amount of data –** The amount of data encrypted under a cryptographic algorithm with key of a given size may be limited by the security policy of the DSN. Similarly the *number of uses* of the key may be limited by the security policy. Such policies typically exist to limit the amount of information related to a specific key available to an adversary for cryptanalysis and to limit the exposure in the case of the compromise of a single key.

- **Time -** The length of time that a key may be used from when it was first used or created may be limited by the policy of the system. Such policies typically exist to limit the amount of time available to the adversary for cryptanalysis and to limit the exposure in the case of the compromise of a single key.
- **Suspected compromise –** A key (long term or session key) may be compromised during pre-deployment or operational phases of a DSN. Key establishment processes may provide two security services that reduce the impact of such compromises. These services are:
  - **Perfect forward secrecy –** The compromise of long term keys does not compromise past session keys, only future session keys are at risk, also known as break back protection.
  - **Known-key attack protection –** The compromise of past session keys does not allow an adversary to corrupt future session keys.
- **Membership changes –** DSN nodes will fail over time for a number of reasons, including node death due to energy depletion. Those nodes that have a pairwise association with the failed node should destroy the corresponding session keys. Groups that include the failed node should replace their group session keys so that if an adversary later compromises the dead node, group traffic will not be compromised. Nodes that detect that they are dying should delete all stored keys. Group session keys should also be changed when a new node is added to a group so that if the new node has been taken over by an adversary past group traffic will be protected.

3.5 Scalability

DSNs have on the order of 10 to 10,000 nodes, of which at most a small number (< 10) of these nodes are energy rich super nodes or gateway nodes. Large DSNs cannot utilize a keying scheme that has poor scaling properties (either in terms of energy cost or latency) for establishing and maintaining a key for the DSN as a whole or for some large subset of nodes.

Most group keying schemes have some cost related parameter (number of encryption operations, number of bits received) that grows rapidly with increasing group size. For lightly used groups,[9] or groups where the members often modify messages rather than just forwarding them, it is more efficient to use multiple smaller subgroups (with different group keys), and simply re-encrypt messages when they are forwarded from one subgroup to another. This approach is especially attractive when transmission energy costs are more important than computational costs, as in the case of the WINS nodes. The re-encrypting cost can be reduced for long messages (assuming key initialization and switching can be done efficiently) by using key enveloping / encapsulation techniques. Such techniques encrypt the message with a single key, K_1, and then re-encrypt that key with another key, K_2. Thus, only K_1 would have to be re-encrypted and not the much larger message.

3.6 Availability

Key management services must ensure that confidentiality and group-level authentication services are available to authorized parties when needed, protecting against active attacks that attempt to interrupt service within the network. To ensure the availability of message protection, the SAFEMITS network should protect its resources (i.e., SAFEMITS nodes) from the unnecessary processing of key management messages in order to minimize energy consumption and extend the life of the network. Key management functions should not limit the availability of the network and

[9] Group whose members send few messages (or total number of bits) that require the use of the group's key over the group's (or individual group key's) lifetime.

not create single points of failure such as a centralized key management node for all network-wide security. The following should be observed:

- The SAFEMITS network should protect its resources (i.e., SAFEMITS nodes) from unnecessary processing in order to minimize energy consumption;
- Security mechanisms within the network should not adversely restrict the availability of SAFEMITS data or inhibit the SAFEMITS network from performing its mission;
- Security mechanisms should not present a "single point of failure" within the network (e.g., should not have a single centralized key management node); and
- Security mechanisms should minimize latency in forwarding data and establishing data protection services (i.e., establishing and supporting key material among SAFEMITS nodes).

The requirement of security not interfering with the operations of the network is important in maintaining the availability of the network. If for some reason the SAFEMITS nodes are not cryptographically synchronized where all SAFEMITS nodes have the proper key material for communication, the availability of the network could suffer. In mission critical scenarios, failure to establish keys between communicating SAFEMITS nodes cannot be tolerated. Therefore, the SAFEMITS network must be able to establish keying relationships in all scenarios, even if a temporary reduction in security is necessary to do so.

3.7 Accessibility

End-to-end confidentiality of SAFEMITS data should not be performed since it prevents SAFEMITS data fusion by intermediate nodes from taking place. An effective technique to extend SAFEMITS network lifetime is to limit the amount of data sent back to reporting nodes. Limiting communicated data reduces communications energy consumption. To maintain a commensurate amount of information while limiting communicated data, some processing of the raw data to discard extraneous or duplicative reports is necessary. This processing requires that intermediate SAFEMITS nodes along the multi-hop communications path between the communications node and the final destination have access to the protected data to perform data fusion processing. End-to-end confidentiality of SAFEMITS data should not be performed.

To provide intermediate node accessibility, a key management scheme must establish keying relationships, either directly or transitively, with all potential intermediate nodes between all potential communications nodes and all potential destination nodes. Direct keying relationships between all potential communications, intermediate, and destination nodes may be accomplished by having a single network-wide key for all nodes. Transitive keying relationships allow intermediate nodes along the multi-hop communications path to decrypt and verify received data via one key, and use another key to re-encrypt and authenticate data to be forwarded. Instead of creating a single network-wide key, transitive relationships allow much smaller groups to establish keying relationships.

3.8 Self-Organization

As distributed SAFEMITS networks must be self-organize their routing, they must also self-organize their key management. It often will not be known prior to deployment where, within the anticipated territory in which the DSN will operate, a particular node will be located. The immediate neighboring nodes of any DSN node will not be known in advance in most circumstances, and in general the number of neighbors, the distances or power required to send a messages with a particular error rate from one node to another will not be known in advance. The location of and distance (physical and number of hops) to the gateway or other special nodes will also not be known a-priori. As a consequence, the DSN nodes must be able to select the appropriate keying mechanism for the situation. The nodes may also have to augment the group keying protocols with other protocols,

such as a protocol for electing a (revolving) group leader or a protocol ordering the nodes that make up the group in order to take advantage of efficiencies of certain group keying mechanisms.

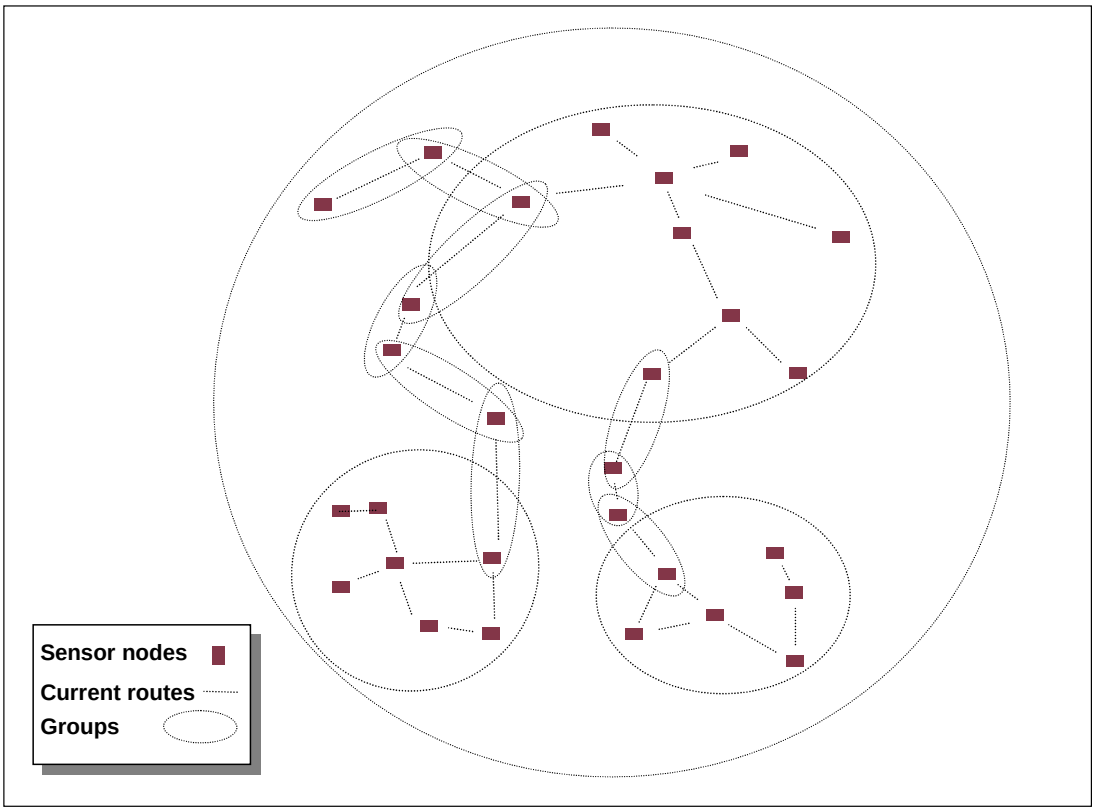

Figure 7 - Establishing Keys Between Small Groups vs. the Entire SAFEMITS Network

The self-(re)organizing capability of DSNs must also be able to deal with nodes failing (or losing contact) during deployment or at other times during the lifetime of the network. These failures may be caused by energy exhaustion, adversary actions (e.g. jamming, artillery barrages, and capture) or through natural causes. If such events require re-keying the affected group keys, then DSN key management (including the key schemes used) must able to handle these events efficiently. Since DSNs self–organize, initially no route will be known between nodes of the network, and even after the routes are initially established they will change due to factors such as changing node energy reserves and the noisiness of communication links that construct the routes. The keying scheme for the DSN must efficiently provide the necessary level of confidentiality and authentication to allow the DSN to operate correctly in such an environment.

3.9 Flexibility

SAFEMITS networks will be used in dynamic battlefield scenarios where environmental conditions, threat, and mission may change rapidly. Changing mission goals may require SAFEMITSs to be removed from or added to an established SAFEMITS node. Furthermore, two or more SAFEMITS networks may be fused into one, or a single network may be split in two. Key establishment protocols must be flexible enough to provide keying for all potential scenarios a SAFEMITS network may encounter. Protocols that require knowledge of what other nodes will be co-deployed are discouraged, whereas protocols with minimal preconceptions are encouraged.

4 Constraints

Having defined requirements for key management, this section focuses on identifying constraints of the distributed SAFEMITS network that may affect the implementation of key management mechanisms. We classify constraints as either SAFEMITS node constraints (Section 4.1) or networking constraints (Section 4.2).

4.1 SAFEMITS Node Constraints

The capabilities and constraints of SAFEMITS node hardware will influence the type of security mechanisms that can be hosted on a SAFEMITS node platform. We assume that most SAFEMITS nodes are inexpensive, limited-capability, generic SAFEMITS nodes. However, there will exist small numbers of greater-capability, energy-endowed gateway nodes that provide either local bridging between sub-networks or clusters or between networks using long-haul circuits.

4.1.1 Battery Power/Energy

Energy is perhaps the greatest constraint to SAFEMITS node capabilities. We assume that once SAFEMITS nodes are deployed in a SAFEMITS network, they cannot be recharged. Therefore, the battery charge taken with them to the field must be conserved to extend the life of the individual SAFEMITS node and the entire SAFEMITS network. Various mechanisms within the network architecture, including the SAFEMITS node hardware, take this limitation into account. When considering implementing a cryptographic function or protocol within a SAFEMITS node, the impact on the SAFEMITS node's available energy must be considered.

When applying security within a SAFEMITS node, we are interested in the impact that security has on the lifespan of a SAFEMITS (i.e., its battery life). The extra power consumed by SAFEMITS nodes due to security is related to the processing required for security functions (e.g., encryption, decryption, signing data, verifying signatures), the energy required to transmit the security related data or overhead (e.g., initialization vectors needed for encryption/decryption), and the energy required to store security parameters in a secure manner (e.g., cryptographic key storage). Since the amount of additional energy consumed for protecting each message is relatively small, the greatest consumer of energy in the security realm is key establishment.

4.1.1.1 Computational Energy Consumption

The amount of computational energy consumed by a security function on a given microprocessor is primarily determined by the processor power consumption, the processor clock frequency, and the number of clocks needed by the processor to compute the security function. The cryptographic algorithm and the efficiency of the software implementation determine the number of clocks necessary to perform the security function. For cryptographic processing, we assume that energy consumption cannot be significantly reduced via a reduction in clock frequency, since a corresponding reduction in voltage would be required, a capability not widely available in today's embedded processors.

Public key cryptographic algorithms such as RSA are computationally intensive, executing thousands or even millions of multiplication instructions to perform a single security operation. Thus, a microprocessor's public key algorithm efficiency is primarily determined by the number of clocks required to perform a multiply instruction. Table 4 shows the wide variance of energy consumption for representative embedded microprocessors in computing a basic public key algorithm building block - a multiply function with a 128-bit result.

Processor	Power Consump-tion (mW)	Clock Freq. (MHz)	Native Mult. Result	# clocks to compute 128-bit result	Time required (µs)	Energy consumed (nJ)
MIPS R4000	230	80	128	40	0.50	115
SA-1110 "StrongARM"	240	133	64	60	0.45	108
Z-180	300	10	32	912	91	27000
MC68328 "DragonBall"	52	16	32	1920	120.	6200
MCF5204 "ColdFire"	625	33	32	304	9.2	5800
MMC2001"M-Core"	81	33	32	416	12.6	1020
ARC 3 [10]	2	40	32	168	4.2	8.4

Table 4 - Computation Time and Energy Consumption for 128-bit Multiply Result

We used the following assumptions to compute the results of Table 4:

- the power consumption values used were taken from the maximum power consumption values for each processor when available, to reflect the fact that the processor usually consumes its maximum power when performing multiplier core operations, and
- the estimate of the number of clocks to compute the 128-bit multiply result includes estimates of the number of clocks to add the result to an accumulator, update the loop counters, and perform other house-keeping such as incrementing and/or decrementing memory pointers.

Symmetric encryption/decryption algorithms and hashing functions consume much less computational energy than public key algorithms. Our estimates of computational energy consumption for AES symmetric encryption and SHA-1 hashing algorithms are shown in Table 5.

Processor	AES[11] Encrypt/Decrypt Energy per bit (mJ/bit)	SHA-1 Hash Energy per bit (mJ/bit)
MIPS R4000	0.000009	0.0000072
MC68328 "DragonBall"	0.000101	0.0000410

Table 5 - AES and SHA-1 Computational Energy Consumption Estimates

To lend perspective on the computational energy consumption rates of Table 5, we note that SAFEMITSia's WINS NG RF subsystem, when transmitting at 10 kbps with 10mW of power, consumes a whopping 0.021 mJ/bit. Thus, the transmission energy consumption rate is over three orders of magnitude greater than the energy consumption rates for encryption and hashing. Similarly, the receive subsystem consumes 0.014 mJ/bit when receiving at the 10 kbps rate.

4.1.1.2 Communications Energy Consumption

In addition to consuming energy through computational processing, security functions also consume energy due to the communication of information between SAFEMITS nodes. Communications energy consumption attributable to security includes:

- exchange of key management information, including encrypted keys, certificates, and nonces; and

[10] Simulation results.

[11] Performance estimated from an average of the Rijndael and Twofish AES finalists.

- per-message additions, including initialization vectors (IVs), encryption padding, authentication tags, and signatures.

Exchange of key management information varies widely depending on the key management algorithms, protocols, and the number of participating nodes. Key management algorithms based on symmetric or elliptic curve cryptography require the exchange of fewer bits than RSA, thus consuming less power. Group keying protocols that take advantage of multicast conserve transmit energy consumption. Reducing the number of keying relationships to only local neighbors reduces the amount of information exchanged, and thus the amount of communications energy consumed.

The communications energy consumption costs of per-message additions are dependent on the number of messages are exchanged. Due to both the computational and communication overhead of per-message additions, we expect messages to be comprised of several small-sized packets.

4.1.2 Rechargeability

We assume that once SAFEMITS nodes are deployed in a SAFEMITS network, they cannot be recharged. Therefore, the battery charge taken with them to the field must be conserved to extend the life of the individual SAFEMITS node and the network. Security functions must minimize energy consumption in order to extend SAFEMITS network life.

4.1.3 Sleep Patterns

In order to conserve energy, we assume that SAFEMITS nodes spend a majority of their operational time in low-power *sleep* modes and only awake when required to processes an event (e.g., a tank detected). For this reason, a node's availability within the SAFEMITS network may be limited. This includes its availability to receive cryptographic key updates. In mobile computing environments, PDA devices like the Palm Pilot have low-power modes that are used to conserve energy [Newman98]. Embedded microprocessors like the Motorola's DragonBall used in the wireless Palm Pilot VII have low-power modes that conserve energy.

The result of these sleep patterns is potential unavailability of a node to receive data. In particular, we are concerned about receiving security related commands (e.g., zeroize) and key material. In order to maintain cryptographic synchronization throughout the SAFEMITS network, it is essential that all nodes use the proper cryptographic material when communicating. Failure to maintain or update to the correct keys could isolate a SAFEMITS node from communications with the rest of the network.

4.1.4 Transmission Range

The communications range of SAFEMITS nodes is limited in order to conserve energy. SAFEMITS nodes from SAFEMITSia and Rockwell Collins have variable transmission power from 10 mW to 100 mW allowing the nodes to restrict their transmission range as necessary [Agre00b]. Reducing the transmission power saves SAFEMITS node energy and provides a lower probability of detection. The actual range achieved from a given transmission signal strength is dependent on various environmental factors. We assume that locally SAFEMITS nodes have a transmission range of approximately 100 meters [Mills00]. Long-haul communications capabilities of greater than 1 km are available gateway nodes. In order to support ad hoc networking, we assume that the assignment of gateway nodes is determined at deployment and can be supported by any node in the network. Gateway nodes may contact *relay* points that transmit the signal even further (e.g., over a satellite link).

4.1.5 Memory

SAFEMITS processors require different types of memory to perform various processing functions. ROM or EPROM is needed for storing the general purpose programming such as an embedded operation system, security functions, and basic networking capability. RAM is needed for storing application programs, SAFEMITS data, and intermediate computations. Programmable memory such as EEPROM and FLASH are needed for storing downloaded application code, data between sleep periods.

4.1.5.1 Program Storage and Working Memory

The amount of program storage available for storing security functionality, such as security mechanism implementations, is unlikely to be a constraining factor on security design. Even the most sophisticated cryptographic algorithms can be represented in the tens of kilobytes of memory, whereas the amount of program storage available in ROM, EPROM, or other nonvolatile memory is likely to be in the hundreds of kilobytes or megabytes.

Likewise, the amount of working memory available for security functionality is unlikely to be a constraining factor. Most symmetric encryption and hashing functions can be executed in less than one kilobyte of RAM. Even the more memory-consuming public key functions can be executed in just a few kilobytes of RAM.

To lend perspective, the WINS NG Processor assembly used in the DARPA SAFEMITS program's demonstration contains 8 Megabytes of ROM and 4 Megabytes of RAM. Although the supported Windows CE operating system would consume a generous portion of both the ROM and RAM, it is likely sufficient program storage and working memory would remain to support the relatively meager memory requirements any conceivable security functionality might have. Although cost considerations will likely reduce the amount of memory in production processors as compared to processors used in demonstrations, the declining cost of memory over time indicates larger memories will soon be available at lower costs anyway.

4.1.5.2 Programmable Storage for Security Information

Key management functions often require some form of programmable memory to store long-term symmetric, public, or private keys. Depending on the concept of operations, security architecture, and memory technology, programming may take place during manufacture, during pre-deployment, or even when deployed on the battlefield. For example, a security design might specify programming the trusted public key of a mission commander into all mission SAFEMITS packages prior to deployment. Such a design would be an effective way of having SAFEMITSs verify the legitimacy of mission commands, while not loading the public key during manufacture, which can be more costly in case of a private key compromise.

Supporting programmable memory in some embedded processor configurations may be difficult and/or costly. Cost considerations encourage integration of processor, memory, and other circuitry onto as few chips as possible, preferably a single application-specific integrated chip (ASIC). That is, the embedded processor, RAM, and programmed ROM, will likely be on a single chip in a production SAFEMITS package. However, few fabrication facilities are able to additionally provide programmable memory, such as EEPROM and FLASH, on these ASICs. Although memory technologies and costs have improved rapidly over time, security designers should remain cognizant of the impact of requiring programmable memory in SAFEMITS processors.

4.1.6 Location Communications

The SAFEMITS network environment may not be supportive of satellite location determination technologies like GPS. GPS may not be well suited for environments that shield its signals (e.g., dense foliage, inside a building). Other technologies such as the *Localizer* developed by AEther Wire Location, Inc. enables SAFEMITS nodes to determine relative positioning to other SAFEMITS nodes [Aether95]. Localization could be tied together with a single node's absolute GPS positioning to provide absolute positioning for SAFEMITS nodes. However, we assume that positioning, absolute or relative, is not available in all situations.

Assuming positioning information is unique (i.e., no two SAFEMITSs share the same location), a SAFEMITS node's position can be used for authentication purposes. Position information can also be used to route targeted to targeted geographical areas [Imielinski98]. Geographic routing may be useful in targeting security related commands (e.g., zeroize, rekey) to areas suspect of compromise.

4.1.7 Tamper Protection

Because of their targeted low cost, we assume that tamper protection for SAFEMITS nodes is limited. Tamper protection falls into two categories: active and passive. Active tamper protection can involve the hardware circuits within the SAFEMITS node to protect SAFEMITSive data. Passive mechanisms include those that do not require energy and include technologies that protect a circuit or provide detection (e.g., protective coatings, tamper seals). Because these mechanisms may require extra circuitry that can add cost to a node and consume valuable energy, we assume that active mechanisms will not be typically found in SAFEMITS nodes. Instead, we assume that passive techniques are more indicative of SAFEMITS node technology.

Tamper protection techniques cannot protect against all attacks. Thus, when designing the SAFEMITS network security architecture, we must assume that one or more SAFEMITS nodes within the network may be compromised. Due to the lack of tamper protection available to SAFEMITS nodes, we assume that a sufficiently capable adversary can extract compromising cryptographic information from a SAFEMITS node. Tamper detection technologies can provide indication that tampering has occurred but have limited value in long-term unattended operations. They can prove useful in detecting tampering prior to deployment (i.e., while in storage) and post deployment.

Since SAFEMITS network missions are typically unattended, the potential for tamper attacks is significant. For this reason, military Type I cryptographic hardware may not be well suited to the SAFEMITS network environment. Type I hardware may contain classified algorithms that may not be suitable for disposable SAFEMITS node technology due to the high risk of compromise. Strong commercial open-source cryptographic algorithms may be acceptable replacements but have yet to be generally approved within the Department of Defense (DoD) to protect classified information. However, National Security Agency (NSA) has made the details of the Skipjack and Key Exchange Algorithm (KEA) algorithms public despite their continued use protecting classified information.

4.1.8 Time

Time within the SAFEMITS network is required for synchronization of events. Time synchronization messages issued from a time source must be resistant to modification attacks in order to maintain network synchronization of events. For example, SAFEMITS node event reports can be time critical. An alteration of their time stamps may change the significant of the report. GPS offers a non-spoofable time source. If GPS is not available, other methods can be used to maintain time within the network. This includes accurate local clocks or network time protocols that synchronize time across a network.

4.1.9 Unattended Operations

Depending on the mission of SAFEMITS network, the SAFEMITS nodes may be unattended for long periods of time. For example, remote reconnaissance missions behind enemy lines may not have any physical contact with friendly forces once deployed. Although they may be managed remotely, we assume that in general SAFEMITS nodes are not in physical contact with ground troops once deployed. This makes it impossible for physical detection of tampering (i.e., through tamper seals) and physical maintenance (e.g., battery replacement). Other maintenance functions are possible (e.g., software updates, key updates) but must be done remotely. The amount of time that a SAFEMITS is left unattended increases the likelihood that an adversary has compromised its key material.

4.2 Networking Constraints

This section discusses constraints specific to distributed SAFEMITS networking. Distributed SAFEMITS networks have unique limitations not encountered in more typical wired LAN environments.

4.2.1 Ad hoc Networking

SAFEMITS networks are ad hoc in nature with the composition of the network determined at the time of deployment. During the SAFEMITS node mission, the composition of the network and its routing topology may change. This constraint limits ability to pre-configure SAFEMITS nodes for specific purposes. SAFEMITS nodes should be able to support various roles in the network to ensure the reliability of the network.

4.2.2 Limited Pre-Configuration

The nature of ad hoc networking requires limited pre-configuration in order to support a flexible and easily deployable network. This constraint limits the amount and type of cryptographic material that should be necessary to deploy a secure SAFEMITS network.

4.2.3 Data Rate/Packet Size

Both the data rate and packet size affect the overall SAFEMITS node energy consumption. We assume that packet sizes within the SAFEMITS network are relatively small, potentially as small as 30 bytes with header [Mills00]. We also assume that the data rates are relatively low, less than 1 kbit/second.

The packet size determines the percentage of overhead in a given message. The message header can be a larger percentage of messages overhead if the message spans packets. Cryptographic services should adhere to packet size restrictions in order to limit the amount of overhead and thus reducing the transmission energy penalty associated with transmitting the extra bits. The low data rate must also be considered when implementing cryptographic services in order to minimize latency throughout the network.

4.2.4 Channel error rate

We assume that low-layer communications protocols will offer error detection and correction services. Errors that propagate into the layers where confidentiality, integrity, or authentication services are applied will affect their verification and authentication processes preventing any application data from being exchanged. In particular, in some modes cryptographic modes of

encryption and decryption, the effects of errors vary depending on the use of feedback or chaining with previous results (e.g., Cipher-Feedback (CFB) mode).

4.2.5 Intermittent connectivity

Intermittent connectivity within the SAFEMITS network may arise from channel fading and the sleep patterns of nodes. We assume that channel fading may be time-dependent and a function of the weather and other battlefield conditions. The sleep patterns of nodes may change over time due to available power and event detection.

The limited availability of SAFEMITS nodes may influence the mechanisms used to reliably distribute security critical messages including cryptographic keying messages and other remote keying messages (e.g., zeroize, CRLs). Because it is a requirement to reliably distribute these types of messages, the reliability mechanisms most overcome intermittent connectivity limitations. Otherwise, cryptographic synchronization issues may result and possible isolate SAFEMITS nodes from the network.

4.2.6 Unreliable communications

We assume that the packet-based routing of the SAFEMITS network is connectionless and thus inherently unreliable. Packets may get damaged due to channel errors or dropped at highly congested nodes. The result is lost or missing packets. Higher network protocols must be introduced to add reliability. Connection oriented transport protocols such as TCP may be added. Reliability is required for the distribution of key material and security critical commands.

4.2.7 Latency

The multi-hop routing of the SAFEMITS network may introduce delay within the network as packets traverse the network. Congestion and node processing can be a factor to the amount of latency in the network. We assume that latency is minimal, however it may pose synchronization issues if time is a critical component to security services (e.g., authentication).

For critical event reports and cryptographic key distribution, latency should be kept to a minimum in order to insure the timeliness of the data. The acceptance of old event reports could produce unreliable fused reports. The acceptance of old cryptographic keys could create cryptographic synchronization problems in the network that isolate SAFEMITS nodes from communicating securely with other network nodes.

4.2.8 Unicast vs. multicast

Although many communications and routing protocols assume that only unicast communications are used, we more broadly assume that multicast may be available for use by our key establishment protocols.

4.2.9 Unidirectional Communications

We assume that not all communications channels are bi-directional. In some cases, unidirectional channels may exist where a SAFEMITS node is only capable of transmitting or receiving data but not both. For example, a SAFEMITS node that is in an active SAFEMITS field where targets are being detected (e.g., enemy tanks) may be collecting and processing data but not transmitting to avoid detection. In this case, the SAFEMITS node may receive data but will not transmit until the danger of detection has subsided. Environmental or adversarial jamming may also cause communications links to be unidirectional.

Unidirectional may impact the design of energy efficient cryptographic key distribution protocols in which energy intensive processing is shared between parties. Instead, one party would assume the bulk of the computations.

4.2.10 Isolated subgroups

Because of intermittent connectivity due to sleep patterns, unidirectional communications, etc., subgroups of a SAFEMITS network may become isolated. These isolated subgroups may not be capable of receiving data such as security critical commands or key material. The subgroups may be only temporarily isolated from the network and may rejoin once they awake or routing tables change. Prior to the establishment of a routing infrastructure, the distributed SAFEMITS network will consist of subgroups that are merging into larger structures.

4.2.11 Frequent Routing Changes

As the available energy decreases in key nodes throughout the network, the need to change the routing topology to balance the energy usage within the network becomes important. Frequent routing changes can mean that the intermediate nodes processing data for and end-to-end session can change. Also, since many security services instead will be provided on a hop-by-hop basis, cryptographic key establishment will occur with local neighbors in the routing topology. If the routing changes, the set of local neighbors may change and thus cryptographic key establishment may need to occur again.

4.2.12 Population Density

The population density of SAFEMITS nodes in the network may vary depending upon the application (e.g., boundary defense, surveillance), the communications capabilities of the SAFEMITS nodes, and the environment (e.g., desert, rain forest). We assume relatively short spacing to provide low-probability of interception (LPI) and to provide energy efficient and strongly connected routing topology. We assume that a typical distance between nodes is less than 100 meters [Mills00]. A typical SAFEMITS network will contain less than 1000 nodes and typical clusters sizes (i.e., when using a network clustering algorithm as in [Mills00, Wang00a]) will be less than 10 nodes.[12]

4.2.13 Unknown Recipients

When a packet is routed through the SAFEMITS network, the packet's source may not know the path the packet takes to its final destination if the packet traverses multiple hops. For this reason, a node may assume that once the packet is transmitted, the intermediary nodes are unknown and may be untrustworthy. For this reason, security services may be applied at either an end-to-end or on a hop-by-hop basis, depending on the SAFEMITSivity and type of data exchanged.

[12] Conversation with Dr. Diane Mills of Lockheed Sanders on 17 February 2000.

5 Keying Protocols

5.1 Background

In Section 5, each key management protocol is examined based on its ability to satisfy distributed SAFEMITS network functionality and security requirements, while efficiently overcoming battlefield constraints. For each protocol, the inability to satisfy any of the requirements described in Section 3 will be noted. Similarly, the inability to overcome any of the constraints described in Section 4 will be noted.

The most important SAFEMITS network constraint posed in Section 4 is that of energy consumption. The amount of energy consumed by key management may be minimized for the total system, minimized for each node, or limited to a maximum for each node. Choosing which metric to optimize determines the most energy-efficient key management approach for a given scenario. For most of the key management protocols in Section 5, we will analyze the average energy cost for each SAFEMITS node participating in the key establishment. Both computational and communications energy consumption values will be presented to distinguish between the two (major) sources of energy consumption.

This section begins with a general discussion of keying techniques and is followed with an analysis of sources of key management protocol energy consumption. Section 5.2 examines pre-deployed keying methods which provide for key establishment between SAFEMITS node without the exchange of messages if the participants know the identities of their peers.. Section 5.3 discusses cryptographic protocols that require the active participation of a special node such as a super node. In this section two new protocols Identity-Based Symmetric Keying and Rich Uncle, along with well-known protocols such as Kerberos are analyzed. Section 5.4 discusses cryptographic protocol that requires the active participation of no special nodes such as the Cliques Group Diffie-Hellman protocol and the Burmester-Desmedt broadcast conference keying protocol. Section 5.5 provides an overall comparison of the techniques.

5.1.1 Key Establishment Steps

Establishing a cryptographic key between two or more participants requires two basic steps: (1) establishing trust between the participating entities, and (2) performing the cryptographic key computation. Both steps have unique requirements for maintaining key confidentiality, providing sufficient authentication and integrity protection, providing availability, etc.

Trust establishment may be accomplished by using either public key or secret-key based mechanisms. Conventional public key mechanisms use digital signatures and public key certificates, which are generated, distributed, and maintained by public key infrastructures. The main advantage of using public key algorithms is resistance to exploitation since each node's public/private key-pair is unique. This uniqueness provides the SAFEMITS network with the opportunity to identify a malicious adversary's attempt to establish an excessive number of keying relationships with legitimate nodes. The disadvantages of this approach include the computational energy consumption of public key algorithms, the communications energy consumption of exchanging public key certificates, and the communications energy consumption of exchanging key relationship information necessary to detect adversaries masquerading as legitimate SAFEMITS nodes.

Using secret-key mechanisms for trust establishment greatly reduces SAFEMITS node energy consumption. Secret-key algorithms can be used to provide trust establishment by authenticating exchanged key material using a key common to the participants. This common key is used to compute a keyed message authentication code (MAC) over the exchanged key management

information, authenticating the fact that the message was sent by a "legitimate" SAFEMITS node. Hash-based MACs (HMACs) such as HMAC-SHA-96 [RFC2404] are suitable for this purpose.

One solution, loading and computing HMACs based on a network-wide "mission" key, guarantees that all SAFEMITS nodes can authenticate and verify exchanged key material. The main advantage of this solution is that HMACs are orders of magnitude more energy efficient than public key algorithms. However, this approach is weak since the compromise of only a single SAFEMITS node will divulge the network-wide key, allowing an active adversary to establish keys with a large number of SAFEMITS nodes without being detected by information available at the security layer. Although algorithms for detecting malicious behavior could be performed at higher protocol layers, such as when fusing SAFEMITS data reports, we believe it is problematic to differentiate malevolent and simply erroneous data in the distributed SAFEMITS network environment.

Similarly, cryptographic key computations will use public key or secret-key algorithms, or a combination of both. Public key and granular secret-key-based protocols are desirable due to the limited scope of the established keys. Network-wide or widely used common keys are undesirable due to their greater vulnerability to compromise and larger body of data encrypted under a single key. Nonetheless, secret-key-based protocols are desirable since they consume less energy than public key-based protocols.

5.1.2 Basic Keying Techniques

Providing key management for confidentiality and group-level authentication is difficult due to the ad hoc nature, intermittent connectivity, and resource limitations of the distributed SAFEMITS network environment. The following sections describe key management protocols that balance security and energy constraints in support of these services. These key management protocols can be categorized as pre-deployed, arbitrated, and self-enforcing autonomous keying protocols.

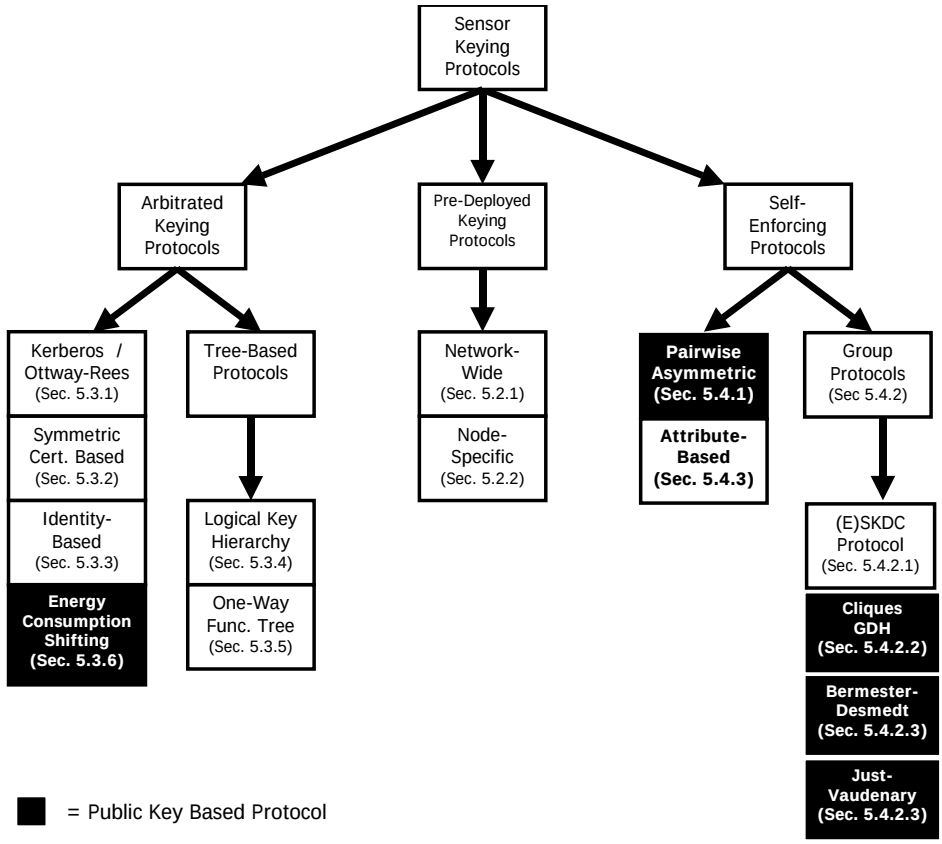

Figure 8 - SAFEMITS Network Keying Protocols

Pre-deployed keying protocols (see Section 5.2) attempt to defray the high SAFEMITS node transmission costs through a more intensive initial pre-configuration. Some pre-configuration is always necessary but can reduce flexibility and impact security. Other techniques require less initial configuration. Arbitrated keying protocols (see Section 5.3) employ a centralized key distribution point to establish and maintain keys for a SAFEMITS network. The central point can be a single centralized entity or distributed among trusted SAFEMITS nodes (e.g., cluster heads). Energy consumption for centralized key distribution is typically localized with the center performing most of the work; however, the use of asymmetric cryptographic protocols can possibly lessen this cost by distributing the computational energy away from the center. Also within this category, hierarchical keying protocols can provide a means of efficiently maintaining the "freshness" of the key material for a group. Self-enforcing autonomous keying protocols (see Section 5.4) distribute the establishment of keys throughout the group, sometime in a pairwise fashion.

5.1.3 Energy Consumption of Keying Primitives

The key management approaches described in this report require the use of cryptographic functions that provide confidentiality, authentication, and integrity. Cryptographic functions can provide these security services serving primitive functions for various key generation and distribution approaches. The selection and placement of the cryptographic functions within the SAFEMITS network influences the energy consumption of individual nodes and thus affect the balance of energy throughout the network. Some functions have symmetric energy costs with the transmitter and receiver of the processed message consuming relatively equal energy (e.g., symmetric cryptographic algorithms). In other cases, energy consumption is asymmetric with different energy costs for the transmitter and receiver of the message (e.g., asymmetric public key cryptographic algorithms).

The following sections describe our approach for addressing the energy costs associated with some primitive cryptographic functions. The functions include encryption/decryption using both symmetric and asymmetric algorithms, digital public key signatures, and hashing functions for authentication and integrity services. These calculations can be applied directly to various embedded microprocessors to determine the energy that may actually be consumed by a SAFEMITS node when implementing the cryptographic mechanism.

5.1.3.1 Energy Computations

In light of the energy constraints for SAFEMITS nodes, it is important to consider the computational energy costs of security functions when implementing them in a SAFEMITS environment. The amount of energy consumed by a security function on a given microprocessor is primarily determined by the processor power consumption, the processor clock frequency, and the number of clocks needed by the processor to compute the security function. The cryptographic algorithm and the efficiency of the software implementation determine the number of clocks cycles necessary to perform the security function.

The following sections describe the energy consumption of primitive cryptographic functions used throughout the key management approaches described in this paper. Core cryptographic functions include public key algorithms (e.g., RSA, DSS), symmetric algorithms (e.g., AES), and integrity/authentication algorithms (e.g., HMACs). The energy required to perform a given operation is computed for various low-power embedded microprocessors typical of SAFEMITS nodes and other wireless commercial devices (e.g., cell phones, PDAs). For example, the MIPS R4400, a 64-bit RISC microprocessor, and the 16-bit Z-180 microcontroller are used in the SAFEMITSia Corporation's WINS NG SAFEMITS node. The SA-1110 StrongARM CISC microprocessor is found in the Rockwell Science Center's WINS SAFEMITS node. Other embedded microprocessors such as the Motorola DragonBall (MC68328), M-Core (MMC2001), and Cold Fire (MCF5204) are found in various low-power consumer electronics that share similar constraints similar to SAFEMITS nodes (e.g., limited battery charge).

Public Key Computations

Public key algorithms provide both confidentiality and authentication services. Because of their high computational costs, they are typically reserved for authentication or encryption of small messages. Algorithms like *RSA* and the *Digital Signature Standard (DSS)* have asymmetric designs whose energy costs vary significantly between signature/verify and encryption/decryption operations.

The computational cost of public key functions is directly related to the costs to perform basic modular arithmetic functions. A processor's total energy consumption for an RSA security operation may be computed from the energy consumption values shown in Table 12. For RSA encryption or verification, the processor must compute a modular exponentiation operation of the form:

$$M^e \bmod n$$

where M is the message being encrypted or verified, e is a public exponent such as 65537, and n is a modulus of at least 1024 bits in size. To compute the number of 128-bit multiply operations necessary for the RSA encryption or verification computation, we assume:

- the Montgomery multiplication method is used;
- the cost of converting into and out of Montgomery space is negligible;
- e = 65537; and
- n is 1024 bits in size;

Thus, with these assumptions, the number of 128-bit operations for the RSA encryption is:

$$\begin{aligned}\#\text{ of 128-bit operations} &= (\#\text{ of 1024-bit modular squares}) * (\#\text{ of 128-bit multiply result operations per 1024-bit modular square}) + (\#\text{ of 1024-bit modular multiplies}) * (\#\text{ of 128-bit multiply result operations per 1024-bit modular multiply})\\ &= \text{Floor}(\log_2(e))*[1.5*(\text{size of } n/64)^2 + 1.5*(\text{size of } n/64)] + 1*[2*(\text{size of } n/64)^2 + (\text{size of } n/64)]\\ &= 16*[1.5*(1024/64)^2 + 1.5*(1024/64)] + 1*[2*(1024/64)^2 + (1024/64)]\\ &= \underline{7056}\end{aligned}$$

For RSA decryption or digital signature, the processor must compute a modular exponentiation operation of the form:

$$M^d \bmod n$$

where M is the message being encrypted or verified, d is the private exponent, and n is a modulus of at least 1024 bits in size. To compute the number of 128-bit multiply result operations necessary for the computation, we use the same assumptions as for the RSA encryption/verification operation and additionally assume the following:

- four-bit exponent scanning;
- the cost of pre-computing values for four-bit exponent scanning is negligible;
- the computation makes use of the Chinese Remainder theorem; and
- d is 1024 bits in size.

Thus, with these assumptions, the number of 128-bit operations for the RSA decryption is:

$$\begin{aligned}\#\text{ of 128-bit operations} &= 2*\{\text{Floor}(\log_2(d/2))*[1.5*((\text{size of } n)/2/64)^2 + 1.5*((\text{size of } n)/2/64)] + (1/4)*\text{Floor}(\log_2(d/2))*[2*((\text{size of } n)/2/64)^2 + ((\text{size of } n)/2/64)]\}\\ &= 2*\{512*[1.5*(512/64)^2 + 1.5*(512/64)] + 128*[2*(512/64)^2 + (512/64)]\}\\ &= \underline{145,408}\end{aligned}$$

The cost of computing the Digital Signature Algorithm (DSA) is computed in a similar fashion with the bit size of each function shown in brackets. The cost of the required SHA-1 hashing is considered negligible.

$$\text{DSA\_Signature}_{[1024]} \approx [1024]^{[160]} \bmod [1024]$$

$$\text{DSA\_Verify}_{[1024]} \approx 2*([1024]^{[160]} \bmod [1024])$$

The cost of computing a Diffie-Hellman operation is computed in a similar fashion with the bit size of each function shown in brackets.

$$\text{DH\_Operation}_{[1024]} \approx [1024]^{[256]} \bmod [1024]$$

The digital signature and verify functions of the *ElGamal* cryptographic algorithm can also be represented in terms of modular arithmetic functions. The ElGamal verification function includes both the cost to generate the two components $g^{M} \bmod(p)$ and $y^{a} a^{b} \bmod(p)$.

$$ElGamal\_Signature_{[1024]} \approx DSA\_Signature_{[1024]}$$

$$ElGamal\_Verify_{[1024]} \approx (13.5)*DSA\_Signature_{[1024]}$$

Lenstra and Verheul [Lenstra00] describe XTR as a method to reduce the number of bits and subsequent cost of modular exponentiation functions. The authors assume that XTR with its *P=Q=170 bits* offers approximately the same security as 1020-bit RSA with a 32-bit public exponent for signature and decryption functions. For encryption and verification functions, we estimate the XTR performance by scaling the performance offered by Lenstra and Verheul to the RSA performance we calculated, modified by our using an RSA public exponent *e* equal to *65537* rather than a full 32-bit public exponent as they suggest.

$$XTR\_Signature_{[1024]} \approx (1360/11900) * RSA\_Signature_{[1024]}$$

$$XTR\_Verify_{[1024]} \approx (2754/500) * RSA\_Verify_{[1024]}$$

$$XTR\_Encrypt_{[1024]} \approx (2720/500) * RSA\_Encrypyt_{[1024]}$$

$$XTR\_Decrypt_{[1024]} \approx (1360/11900) * RSA\_Decrypt_{[1024]}$$

Algorithms based on elliptic curve cryptography (ECC) were not evaluated in this draft since reliable performance numbers on embedded processors could not be found in the public domain. Although using ECC algorithms in place of algorithms such as RSA and Diffie-Hellman may provide significant reductions to energy consumption, Lenstra and Verheul [Lenstra00] contend that XTR provides even greater improvement.

Table 6 compares the relative energy costs for these public key algorithms for various embedded microprocessors. These costs are based on the performance of modular exponentiation functions within each microprocessor and the chip's maximum power consumption.

Processor	Clock Speed (MHz)	Max. Power Load (mW)	Computational Energy Consumption (mJ)								
			RSA Sign	RSA Verify	DSA Sign	DSA Verify	Diffie-Hell-man	El Gamal Sign	El Gamal Verify	XTR Sign	XTR Verify
MIPS R4000	80	230	16.7	0.81	9.9	20.	15.9	9.94	134	1.91	4.5
SA-1110 "StrongARM"	133	240	15.0	0.74	9.1	18.2	14.6	9.1	123	1.71	4.1
Z-180	10	300	3700	184	2300	4500	3640	2300	31000	420	102
MC68328 "DragonBall"	16	52	840	42	520	1040	829	520	7000	96	230
MCF5204 "ColdFire"	33	625	775	39	480	960	765	480	6500	89	214
MMC2001 "M-Core"	33	81	137	6.9	85	169	136	85	1140	15.7	38.00
ARC 3 [13]	40	2	1.13	0.06	0.70	1.40	1.12	0.70	9.4	0.13	0.31

Table 6 – Computational Energy Costs for Public Key Authentication Algorithms

Encryption/Decryption Computations

The National Institute of Standards and Technology (NIST) is sponsoring the development of a next generation of cryptographic algorithms called the *Advanced Encryption Standard (AES)*[14]. We consider these algorithms for their confidentiality services (i.e., encryption, decryption). As a conservative estimate of AES processing, we based our calculations on results from [Aoki00], who compares encryption times for various AES candidate algorithms. The algorithm finalists in Round 2 of the AES selection process include MARS, RC6, Rijndael, Serpent, and Twofish. Of these five, four algorithms have shown encryption times of less than 400 processor clock-cycles for 128-bit encryption on a 32-bit microprocessor like the Intel Pentium II [Aoki00]. Based on these results, we assume a conservative estimate of 400 clock-cycles to perform a 128-bit block encryption with a 128-bit key on a 32-bit microprocessor like the MIPS R4400. We scaled these performance numbers to other embedded processors shown in the following table by considering the processors registers sizes and instruction execution times (i.e., move from register to memory, add, shift/rotate, and XOR). The time that each operation can be completed in is also affected by the size of the registers - using smaller registers will require more time to perform the same operation. We estimate that the cost of decryption for these symmetric algorithms is roughly equivalent to their encryption costs.

[13] Simulation results.

[14] http://www.nist.gov/aes

Processor	Scaling Factor	AES Energy Consumption (mJ/128-bit block)
MIPS R4400	1	0.00115
SA-1110 "StrongARM"	3	0.00217
Z-180	20	0.24
MC68328 "DragonBall"	10	0.0130
MCF5204 "ColdFire"	5	0.038
MMC2001"M-Core"	3	0.00295
ARC 3 [15]	4	0.00008

Table 7 - AES Energy Consumption Estimates

The resulting energy consumption of symmetric AES cryptographic algorithms is significantly lower than asymmetric public key algorithms and is often outweighed by asymmetric cryptographic functions (e.g., RSA) or transmission costs. For example, to encrypt a 1024-bit block consumes approximately 42 mJ on the DragonBall processor using RSA while only 0.104 mJ using our estimation of an AES algorithm. In comparison, the transmission costs for a 1024-bit message are roughly 21.5 mJ and 14.3 mJ for transmission and reception, respectively – approximately 100 times more than AES encryption.[16]

Integrity/authentication

Because of the significant processing costs of public key authentication functions, an alternative that provides authentication and integrity based on hashing algorithms is a lower cost alternative. *Hash-based Message Authentication Codes (HMACs)* compute a *message authentication code (MAC)* for a message x using a secret key k, a message digest code (MDC) h, and padding p.

$$HMAC(x) = h(k \,||\, p \,||\, x \,||\, k)$$

Using the hashing algorithms SHA-1 and MD5 as the MDC function h, we can estimate the number of MDC functions with required to HMAC a message of length m is approximated by the following formula that accounts for the message size and required padding:

$$\#\ MDC\ Functions = 1 + \lceil (m + 65)/block\_size \rceil$$

The input *block_size* for both SHA-1 and MD5 is 512 bits. Table 9 shows the energy cost to compute the HMAC for a 1024-bit message based on the hashing results from Table 8. The hashing cycles for each embedded processor were scaled in a similar fashion as was done for the AES algorithms, i.e. based processors registers sizes and instruction execution times. In comparison, the cost to transmit and receive a 1024-bit message with SAFEMITSia's WINS NG RF subsystem at 10 kbps with 10mW

[15] Simulation results.

[16] For the SAFEMITSia WINS NG RF subsystem transmitting at 10 kbps with 10mW of power. Reception at 10 kbps.

of power is approximately 21.5 mJ. Similarly, the receive subsystem consumes 14.3 mJ when receiving a 1024-bit message at the 10 kbps rate. The WINS NG transmit/receive costs are more than 1000 times greater than the cost to HMAC a message of equivalent size on the MIPS R4400 and StrongARM processors.

Processor	Scaling Factor	SHA-1 Cycles per byte	MD5 Cycles per byte	SHA-1 Energy per byte (mJ)	MD-5 Energy per byte (mJ)
MIPS R4000	1	20	10	0.000058	0.000029
SA-1110 "StrongARM"	3	60	30	0.000108	0.000054
Z-180	20	400	200	0.012000	0.006000
MC68328 "DragonBall"	10	200	100	0.000650	0.000325
MCF5204 "ColdFire"	5	100	50	0.001894	0.000947
MMC2001"M-Core"	3	60	30	0.000147	0.000074
ARC 3 [17]	4	80	40	0.000004	0.000002

Table 8 - SHA-1 and MD5 Energy Consumption

Processor	HMAC-SHA-1 Energy (mJ)	HMAC-MD5 Energy (mJ)
MIPS R4000	0.0115	0.0058
SA-1110 "StrongARM"	0.0217	0.0108
Z-180	2.4015	1.2008
MC68328 "DragonBall"	0.1301	0.0650
MCF5204 "ColdFire"	0.3790	0.1895
MMC2001"M-Core"	0.0295	0.0147
ARC 3 [18]	0.0008	0.0004

Table 9 - HMAC Energy Consumption Estimates for a 1024-bit Message

5.1.3.2 Impact of Key Management Energy Costs on Routing

The energy and latency costs associated with security functions may have some influence over the selection of routes within the SAFEMITS network. SAFEMITS nodes that act as a center for key distribution to a local group may become overly tasked when asked to perform computationally intensive cryptographic functions that can drain energy and introduce network latency. Some keying protocols take this into consideration by distributing their cryptographic computations across a group and thus the energy to establish and maintain key freshness. Network routing protocol should

[17] Simulation results.

[18] Simulation results.

consider the energy impact of security when attempting the balance network energy reserves to maintain strong network connectivity.

5.2 Pre-deployed Keying

Loading keys into SAFEMITS nodes prior to battlefield deployment offers energy-efficient solutions to providing confidentiality and group-level authentication keys. However, pre-deployed keying can be inflexible to changing mission configurations and poses security concerns. The following sections describe various methods of keying SAFEMITS nodes prior to deployment and examine their ability to satisfy the key management requirements.

5.2.1 Network-Wide Pre-deployed Keying

One of the simplest and most energy-efficient key management methods is pre-deployment of a network-wide key. Prior to battlefield deployment, SAFEMITS nodes are loaded with the same key by a mission authority. Alternatively, key material from which one or more keys are derived can be loaded. Since each member of the network contains the same keying material, confidentiality and authentication keys for SAFEMITS data protection are easily computed without any expensive energy consuming computations or communications.

Unfortunately, pre-deployment of a network-wide key is not very secure in many battlefield scenarios. Unattended SAFEMITS nodes in hostile territory are tempting targets for enemy counter-intelligence operations. Compromise of only a single SAFEMITS node exposes the confidentiality keys of all SAFEMITS nodes in the network, potentially disclosing all future communications as well as all past recorded communications to a passive adversary. Similarly, compromise of the networks authentication keys exposes all future communications to undetectable forgery by an adversary.

Furthermore, pre-deployment of a network-wide key may prevent the network from adding new nodes or participating in coalitions. SAFEMITS nodes added to a network must either have the same key loaded as those of the already deployed nodes, or deployed nodes must somehow be securely instructed to also use another network-wide key. Similar problems occur if two separately deployed networks need to inter-communicate.

Pre-deploying a network-wide key for only authenticating exchanged key management information is an attractive alternative approach. Although the vulnerability of a network-wide key is no less real, the ramifications of compromise are less severe. To exploit the compromise, an adversary uses the key to establish keying relationships with as many legitimate SAFEMITS nodes as possible. When SAFEMITS application data is communicated through the network, any data forwarded to the compromised node is disclosed. Exploitation of this approach is less severe since it does not disclose any past communications, only discloses future communications that pass through the compromised SAFEMITS node, and requires the adversary to actively communicate with legitimate SAFEMITS nodes to establish keying relationships. This last condition exposes the adversary to detection unlike vulnerabilities that can be exploited by a passive adversary.

5.2.2 Node-Specific Pre-deployed Keying

The node-specific pre-deployed keying method pre-computes shared keys off-line for possible combinations of pairs of SAFEMITS nodes and loads the appropriate keys into the nodes prior to their deployment. Once deployed the nodes only need to know the identifier of a peer node in order to communicate securely with the peer. This keying method can be extended to small groups by pre-calculating keys of possible groups of nodes of a certain size or less.

Whenever the Mission Authority cannot anticipate where nodes will be located, to allow for maximum network connectivity and security each Node Y receives keys that will allow it to securely

communicate with all other current DSN nodes and those nodes that will be added during the lifetime of Node Y. These other nodes must also have a key loaded prior to their deployment for Node Y. This keying method has essentially zero energy cost (for the SAFEMITS nodes) and latency for the DSN nodes.

The keying method lacks flexibility and does not scale well. Once a node has been deployed, the set of nodes that it can form an association with is fixed and cannot be extended without employing additional (non pre-deployed) keying techniques. A Mission Authority using the group wide pre-deployment method does not have to provide for future deployments of SAFEMITS nodes when deploying nodes in the present.

The scalability problem stems from the ad-hoc nature of the DSN deployment and operation, which prevents the Mission Authority from reliably anticipating which pairs of DSN nodes[19] will need a key. For a static DSN network of size *N*, the number of keys necessary for forming groups of size *G* < *N* is *N! / G!(N-G)!* The total number of keys necessary for all groups of size *G or less* is:

$$\sum_{g=2}^{G}\binom{N}{g}$$

The total number of keys necessary per node for all groups of size *G or less* is:

$$\sum_{g=1}^{G-1}\binom{N-1}{g}$$

Table 10 shows the total number of keys that need to be generated for groups of size 2, 3, 6, 12 *or less* by the DSN owner and Table 11 shows the amount of memory need by each DSN SAFEMITS node to store its keys, assuming 20 bytes of memory per key.

Number of network nodes	Total Number of Pre-deployed Keys			
	for all possible pairs of nodes	for all possible pairs and triplets	for all groups 6 or less	for all groups 12 or less
50	1.23×10^{3}	2.08×10^{4}	1.83×10^{7}	1.72×10^{11}
100	4.95×10^{3}	1.67×10^{5}	1.27×10^{9}	1.21×10^{15}
500	1.25×10^{5}	2.08×10^{7}	2.13×10^{13}	4.57×10^{23}
1000	5.00×10^{5}	1.67×10^{8}	1.38×10^{15}	2.00×10^{27}
5000	1.25×10^{7}	2.08×10^{10}	2.17×10^{19}	5.04×10^{35}
10000	5.00×10^{7}	1.67×10^{11}	1.40×10^{21}	2.08×10^{39}

Table 10 - Total Number of Keys Required for Node-Specific Pre-deployed Keying

[19] With the same size and deployed within some specified time interval.

Number of network nodes	Storage Required per Node (in bytes)			
	for all possible pairs of nodes	for all possible pairs and triplets	for all groups 6 or less	for all groups 12 or less
50	9.80×10^{2}	2.45×10^{4}	4.28×10^{7}	7.99×10^{11}
100	1.98×10^{3}	9.90×10^{4}	1.51×10^{9}	2.87×10^{15}
500	9.98×10^{3}	2.50×10^{6}	5.11×10^{12}	2.20×10^{23}
1000	2.00×10^{4}	9.99×10^{6}	1.65×10^{14}	4.74×10^{26}
5000	1.00×10^{5}	2.50×10^{8}	5.20×10^{17}	2.42×10^{34}
10000	2.00×10^{5}	1.00×10^{9}	1.67×10^{19}	4.98×10^{37}

Table 11 – Storage Requirements for Node-Specific Pre-Deployed Keying

For pairwise keying, the anticipated maximum size of a DSN (10,000 nodes) can be handled by this technique, if there is no significant node replacement over the lifetime of the DSN. This method alone cannot support groups of three nodes in even a static, medium size DSN network (500-1000 nodes) and is totally unsuited for larger groups in all but trivially sized DSNs.

5.2.3 J-Secure Pre-Deployed Keying

The keying method of Section 5.2.2 is secure against any coalition of compromised nodes whereas the network wide pre-deployed keying method is vulnerable to the compromise of any one node. The J-secure pre-deployed keying methods offer compromise protection between that of the above methods. The J-secure methods can maintain security of subgroups of nodes against coalitions of up to $(1 < j < n)$ compromised nodes that are not part of the subgroup. These methods scale better than the node-specific method but also lack flexibility.

Blom [Blom84] proved that for any J-secure pre-deployed method with m-bit size pairwise session keys the minimum amount of key material that must be stored in each node is $m(j + 1)$ bits. This value translates into $(j + 1)$ keys.[20] In the same paper Blom presented a method for doing pairwise J-secure pre-deployed keying for any $j < n - 2$. We will not present that method here but rather note that using Blom's method we can provide every SAFEMITS node with a key to communicate with every other SAFEMITS node (no matter how large the DSN) using only 2.0 x 10^4 bytes of storage and be protected against the compromise of up to one thousand nodes.

However, like the method of Section 5.2.2, we cannot combine two DSNs that have already been deployed using this method (or other pre-deployed keying methods) unless the nodes were configured anticipating that the two DSNs might be combined. Reconfiguring a DSN after deployment, to support another DSN using pre-deployed keying techniques consumes too much energy.

For any J-secure method with m-bit session keys, providing for all possible groups of nodes of a size t, independent of the DSN of size, requires at a minimum that each nodes stores

[20] The method of Section 5.2.2 meets this bound (with j equal to the DSN size) and therefore cannot be improved upon without sacrificing security.

$$m \cdot \begin{pmatrix} j+t-1 \\ t-1 \end{pmatrix}$$

bits, Blundo *et al.* [Blundo92]. In the same paper the authors provide a method that meets this bound. Table 12 shows the storage requirements for some possible combinations of DSN size, group size and number of compromised nodes that can be tolerated without compromising the security of group of non-compromised nodes.

Number of compromised nodes tolerated	Storage Required per Node (in bytes)			
	for all possible pairs of nodes	for all possible pairs and triplets	for all groups 6 or less	for all groups 12 or less
25	5.20×10^2	7.02×10^3	2.85×10^6	1.20×10^{10}
50	1.02×10^3	2.65×10^4	6.96×10^7	8.36×10^{12}
200	4.02×10^3	4.06×10^5	5.74×10^{10}	1.42×10^{19}
300	6.02×10^3	9.09×10^5	4.26×10^{11}	1.10×10^{21}
500	1.00×10^4	2.52×10^6	5.37×10^{12}	2.79×10^{23}
1000	2.00×10^4	1.00×10^7	1.69×10^{14}	5.35×10^{26}

Table 12 – Storage Requirements for J-Secure Pre-Deployed Keying

For pairwise keying, the anticipated maximum size of a DSN (10,000 nodes) or more nodes can be handled by this technique, *assuming that the level of compromise tolerance of 1000 or less is acceptable.* This method can also support groups of three but the level of compromise that can be provided using a reasonable amount of SAFEMITS node storage (less than 10^6 bytes) is limited. J-secure methods cannot be used for groups of six or more with acceptable compromise tolerance.

The reader should also be aware that all of the techniques in this section only provide a single key for each pair or other small subgroup of nodes. A method for generating session keys will also be needed if pre-deployed methods are to form the basis for establishing long-term security (confidentiality or authentication). Since cryptographic keys that are used for authentication (without non-repudiation) are not SAFEMITSive to the compromise of *expired* keys the above methods can suit the needs of DSNs. These methods differ significantly in their cost, degree of compromise protection and in the impact of a compromise. They all have limited flexibility, though the group-wide keying method is considerably more flexible than the other methods.

5.3 Arbitrated Protocols

A large number of secret-key and public-key based cryptographic techniques have been developed for interactively establishing shared pairwise and group keys. The techniques can be divided into arbitrated protocols (where a trusted server is used as part of the protocol) and "autonomous" protocols where no trusted third party is used. The protocols can be further categorized into secret or public key protocols depending on the dominant mechanism by which the shared key is established, rather than the means by which the participants in the protocol are authenticated.

In this section and the next we will examine representative protocols from the various classes of key establishment protocols. Before we begin looking at these protocols we observe that in general secret-key protocols have substantially lower computational energy requirements and occasionally better communication energy costs than do protocols that rely on public key techniques. Public key protocols are more expensive computationally. The difference in communication costs is due to the

smaller certificates and key sizes use in secret-key protocols. However, secret-key techniques have greater pre-configuration requirements (e.g. the Mission Authority has to generate and store securely many more keys) than do public key based techniques.

5.3.1 Traditional Key Distribution Center-Based Methods

A large number of secret-key based methods have been developed that require an interactive trusted third party, a Key Distribution Center (KDC) or a Key Translation Center (KTC),[21] in order to establish a shared key between any two members of the system. Kerberos [Newman94], Needham-Schroeder [Needham78], Otway-Rees [Otway87], Bellare-Rogaway [Bellare93] are a few of these protocols. In these methods a trusted server shares a unique secret-key with each SAFEMITS node. The KDC or KTC securely stores these shared keys in a local database. These secret keys must be distributed to the SAFEMITS nodes prior to their deployment, which impacts the deployment of new trusted servers in the future since the appropriate secret keys for these servers must be calculated in advance and stored in SAFEMITS nodes. Updating the SAFEMITS nodes after they are deployed consumes significant energy. Reusing the secret keys between servers is an option, with a SAFEMITS node sharing one key with each server, but this approach results low key granularity, each server becomes a single point of failure for the security of the system.

In the following sections we look at two KDC-based protocols, Kerberos and Otway-Rees. The most notable differences between them is the difference in energy consumption and that Kerberos requires that each SAFEMITS node and KDC have secure synchronized clocks, whereas the Otway-Rees protocols do not. We will describe each protocol, their security properties and examine their energy consumption.

5.3.1.1 Kerberos

The Kerberos series of protocols were originally based on the Needham-Schroeder protocol. The Kerberos protocols use a KDC to establish a secret shared key between (in the case of a SAFEMITS network) two SAFEMITS nodes. The four-pass Kerberos protocol provides mutual entity authentication, key confirmation and a key freshness guarantee between the SAFEMITS nodes. Here we present the Kerberos version 5 protocol with certain fields removed for clarity or because they are not needed in a SAFEMITS network environment.

The protocol

Each SAFEMITS node *i* has a secret key that it shares with a KDC (KDC_j). ID_A is the unique identifier of node A, likewise for node B. N_A is a random value called a nonce that is used (with sufficiently high probability) no more than once for the same purpose [Menezes, p397].

Round 1

Node A → Node KDC_j: $KDC_j \,||\, ID_A \,||\, ID_B \,||\, N_A$

The initiator, Node A, sends its identifier and the identifier of Node B which we assume Node A already knows to KDC_j. The KDC looks up Node A and Node B in its database, verifies that they are valid supported nodes, and fetches their corresponding shared keys K_{Aj} and K_{Bj}.

Define $ticket_B$ as $E(K_{Bj}, K_{pair} \,||\, ID_A \,||\, L)$ where K_{pair} is the session key that will be shared by Node A and Node B. L is the lifetime of the key. The KDC generates a ticket for Node A to forward to Node B and also generates an authenticator, $E(K_{Aj}, K_{pair} \,||\, ID_B \,||\, N_A \,||\, L)$, that provides Node A with

[21] A Key Translation Center is a KDC that translates a key securely provided by one party into a form that can be securely transferred to another party.

the shared key and proof that this is the right shared key to use with Node B at this time. The KDC sends the following message back to Node A.

Round 2

Node KDC_j → Node A: $ID_A \,||\, KDC_j \,||\, ticket_B \,||\, N_A \,||\, E(K_{Aj},\ K_{pair} \,||\, ID_B \,||\, N_A \,||\, L)$

Node A decrypts $E(K_{Aj},\ K_{pair} \,||\, ID_B \,||\, N_A \,||\, L)$ and verifies that ID_B and N_A match the message it sent the KDC in Round 1. Then Node A takes a fresh timestamp T_A, creates the Round 3 message and sends it to Node B

Round 3

Node A → Node B: $ID_B \,||\, ID_A \,||\, ticket_B \,||\, E(K_{pair,}\ ID_B \,||\, T_A)$

Node B upon receipt of the message decrypts $ticket_B$ and obtains K_{pair}, and uses it to decrypt $E(K_{pair,}\ ID_B \,||\, T_A)$. Node B can then verify the identifier fields obtained from both encrypted values, that the timestamp T_A is still valid, and that Node B's local time is within the lifetime L. If the verification step succeeds Node B encrypts the timestamp provided by Node A with the shared key and sends the Round 4 message to Node A.

Round 4

Node B → Node A: $ID_A \,||\, ID_B \,||\, E(K_{pair,}\ T_A)$

Node A decrypts the message and verifies that the timestamp is same one it generated for Node B in Round 3. This message allows Node A to determine that Node B received the Round 3 message and successfully decrypted the shared key.

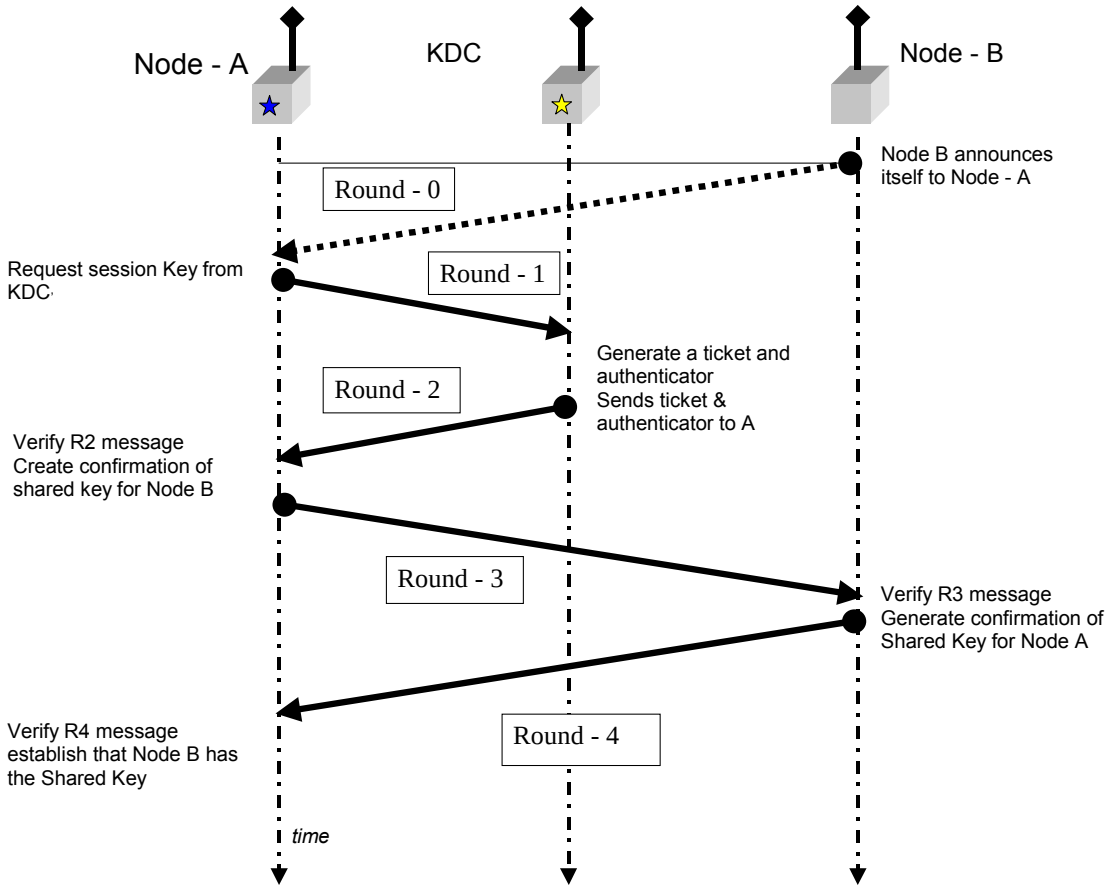

Figure 9 - Kerberos V (modified for DSN use)

Analysis

This version of the Kerberos protocol provides mutual entity authentication, key confirmation and a key freshness guarantee. Each SAFEMITS node performs two secret-key decryptions and one secret-key encryption. Under these assumptions:

- The node ID and KDC ID sizes are 64 bits.
- The symmetric keys are 128 bits.
- All nonces are 64 bits.
- The timestamps and validity periods (lifetimes) are 64 bits.
- The Round 1 message is 192 bits.
- The Round 2 message is 832 bits.
- The Round 3 message is 704 bits.
- The Round 4 message is 320 bits.

The communication energy costs are: 35 mJ for Node A, 17 mJ for Node B, 20 mJ for the KDC. As is shown in Table 13 the computational energy cost of this method is relatively low. The total energy cost of the protocol is essentially the same as the communication cost for all of the processors except for the Z-180, where the total cost ranges from 5% to 10% more than the communication cost for the protocol participants.

Processor	Computational Energy Consumption (mJ)		
	Node A	Node B	KDC
MIPS R4000	0.0081	0.0075	0.0052
SA-1110 "StrongARM"	0.015	0.014	0.0097
Z-180	1.7	1.6	1.1
MC68328 "DragonBall"	0.091	0.098	0.072
MCF5204 "ColdFire"	0.027	0.25	0.17
MMC2001 "M-Core"	0.021	0.019	0.013
ARC-3	5.6×10^{-4}	5.2×10^{-4}	3.6×10^{-4}

Table 13 – Kerberos Protocol Computational Energy Consumption

An important consideration with protocols that use trusted servers is the total energy cost of the protocol (minus the energy rich trusted server). The energy cost of the protocol increases with increasing distance (number of hops and hop distance) between the participants. The above communication energy costs were based on the assumption that Node A was one hop away from Node B and vice versa and we did not include the energy cost imposed on any intermediate nodes between Node A and the KDC. In the next table we display estimated communication energy cost to the DSN of this protocol (minus the KDC cost) for a number of different hop counts between Node A and the KDC, we again assume that Nodes A and B have no intermediate nodes. The total energy cost is essentially the same as the communication energy cost.

Number of Hops	Communication Energy Consumption (mJ)
1	52
2	88
4	130
8	220
16	390

Table 14 – Kerberos Protocol Multi-hop Total SAFEMITS Node Communication Energy Consumption

5.3.1.2 Otway-Rees

The Otway-Rees protocol is also a trusted server-based four-pass protocol. Unlike Kerberos, this protocol only uses nonces rather than timestamps to prevent replay and establish key freshness. Secure synchronized clocks are therefore not used in this protocol. Here we present the protocol with certain fields removed for clarity or because they are not required in a DSN environment.

The protocol

The protocol consists of four rounds. Each SAFEMITS node *i* has a secret key that it shares with a KDC (KDC_j). ID_A is the unique identifier of Node A likewise for Node B. N_A and M are nonces. Nonce M functions as a transaction identifier.

Round 1

Node A → Node B: $KDC_j \,||\, M \,||\, ID_A \,||\, ID_B \,||\, E(K_{Aj}\,,\, N_A \,||\, M \,||\, ID_A \,||\, ID_B)$

The initiator, Node A, generates an encrypted value, $E(K_{Aj}\,,\, N_A \,||\, M \,||\, ID_A \,||\, ID_B)$, for the KDC that identifies the participants in the protocol and include the nonces N_A and M to prevent replay attacks.

Node A sends the Round 1 message to Node B. Node A must know node B's identity prior to sending the message since that input is encrypted under Node A's shared key with KDC_j.

Node B receives the message and creates an encrypted value $E(K_{Bj}, N_A \| M \| ID_A \| ID_B)$ (corresponding to the same transaction as does Nodes A's $E(K_{Aj}, N_A \| M \| ID_A \| ID_B)$). Node B combines this encrypted value with the Round 1 message components and sends the resulting message to the KDC.

Round 2

Node B → Node KDC_j: $KDC_j \| M \| ID_A \| ID_B \| E(K_{Aj}, N_A \| M \| ID_A \| ID_B) \| E(K_{Bj}, N_B \| M \| ID_A \| ID_B)$

Upon receiving the Round 2 message the KDC looks up nodes A and B in its database (checking the nodes are valid) and uses the stored, shared keys to decrypt both encrypted parts of the message. The KDC checks that the node identifiers and the nonce M are used consistently throughout the message.[22] The KDC generates a session key K_{pair} and encrypts it and the nonces provided by nodes A and B under the appropriate session keys. The concatenation of these two encrypted values is sent back to Node B as the Round 3 message.

Round 3

Node KDC_j → Node B: $ID_B \| E(K_{Aj}, N_A \| K_{pair}) \| E(K_{Bj}, N_B \| K_{pair})$

Node B decrypts $E(K_{Bj}, N_B \| K_{pair})$, and verifies that N_B matches the nonce it used earlier. If the values match, Node B uses K_{pair} as its shared key with A and sends the first half of the message to Node A optionally concatenated with both nonces encrypted under the shared key.

Round 4

Node B → Node A: $ID_A \| E(K_{Aj}, N_A \| K_{pair}) \| [E(K_{pair}, N_A \| N_B)]$

Node A decrypts $E(K_{Aj}, N_A \| K_{pair})$, and verifies that N_A matches the nonce it used earlier. If the values match, Node A uses K_{pair} as its shared key with B.

If the optional part of the Round 4 message is sent to Node A, then Node A decrypts it and verifies nonce N_A. Then Node A encrypts N_B using the shared key and sends that result back to Node B.

Round 5 (Optional)

Node A → Node B: $[ID_B \| E(K_{pair}, N_B)]$

Node B decrypts the message and verifies N_B.

[22] Otherwise this protocol is vulnerable to an intruder-in-the middle attack [Boyd93, vanOorshot93].

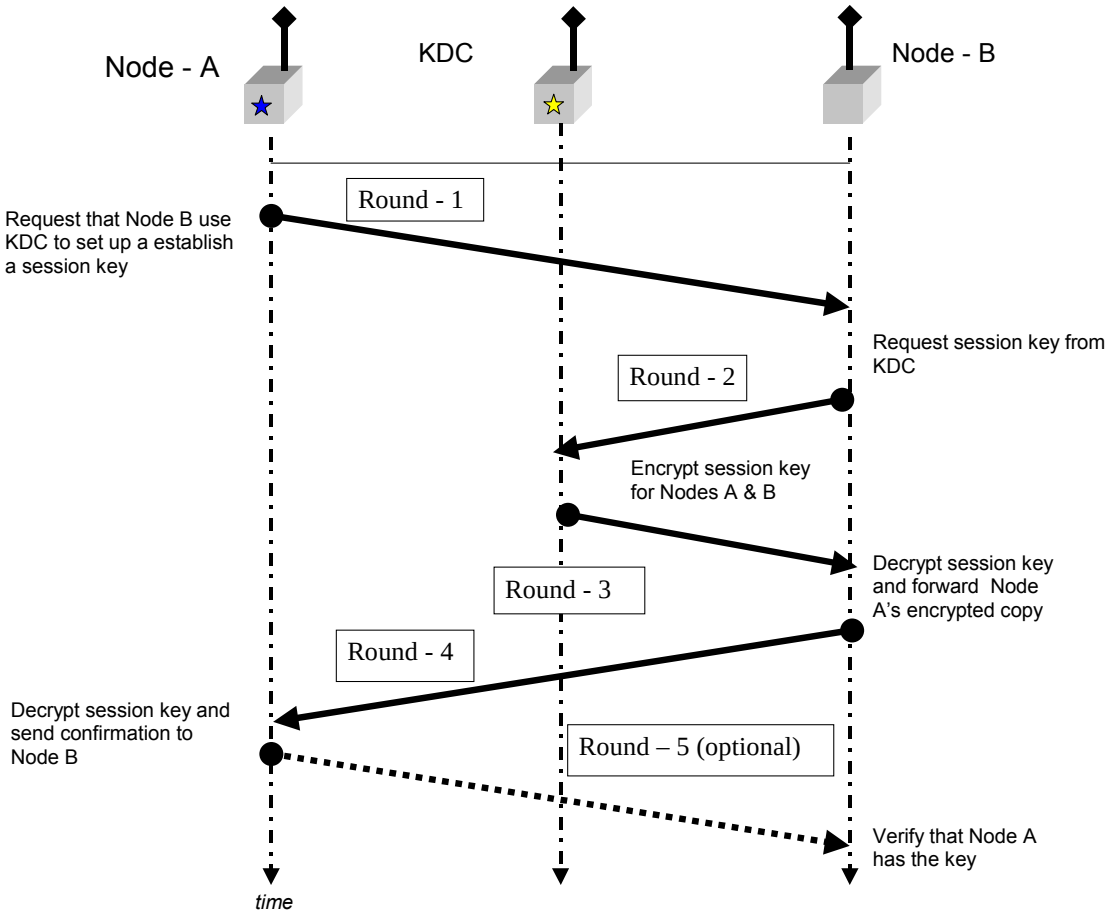

Figure 10 - Otway-Rees Protocol (modified for DSN use)

Analysis

This Otway-Rees protocol provides Nodes A and B with assurance that K_{pair} is fresh. Unless the optional fifth message is used, Node B has limited assurance that Node A knows K_{pair} until subsequent use of the shared key by Node A. Also, using the optional field in Round 3 of the protocol gives Nodes A and B key confirmation and entity authentication assurance.

If the optional parts of the protocols are performed each SAFEMITS node performs a secret-key encryption and decryption, two encryptions and decryptions. Like the Kerberos protocols the energy cost of the Otway-Rees protocol is dominated by the communication energy costs. Under these assumptions:

- The node ID and KDC ID sizes are 64 bits.
- The symmetric keys are 128 bits.
- All nonces are 64 bits.
- The Round 1 message is 512 bits.
- The Round 2 message is 768 bits.
- The Round 3 message is 448 bits.
- The Round 4 message is 384 bits.

The resulting communication energy costs are: 16 mJ for Node A, 38 mJ for Node B, and 20 mJ for the KDC. As shown in Table 15, the computational energy cost of this method is relatively low, though not as low as the Kerberos V protocol.

Processor	Computational Energy Costs (mJ)	
	Nodes A and B	KDC
MIPS R4000	0.0052	0.0081
SA-1110 "StrongARM"	0.0097	0.0152
Z-180	1.08	1.44
MC68328 "DragonBall"	0.059	0.0910
MCF5204 "ColdFire"	0.171	0.27
MMC2001 "M-Core"	0.0133	0.021
ARC-3	0.0004	0.0006

Table 15 – Otway-Rees Protocol Computational Energy Consumption

The communication energy costs were based on the assumption that Node A was one hop away from Node B and vice versa and the energy cost imposed on any intermediate nodes between Node B and the KDC were not included. The next table displays estimated total energy costs to the DSN of this protocol for a number of different hop counts between Node B and the KDC.

Number of Hops	Communications Energy Consumption (mJ)
1	99
2	180
4	260
8	410
16	723

Table 16 - Otway-Rees Protocol Multi-Hop Communication Energy Consumption

Compared to the Kerberos protocol, the Otway-Rees protocol consumes much more energy when the KDC is many hops away from the SAFEMITS nodes. The greatest negative impact of choosing the Otway-Rees method over the Kerberos protocol is the impact on the ordinary SAFEMITS nodes surrounding the presumably energy-rich KDC. Intermediate nodes that handle communications in both directions between the KDC and other SAFEMITS nodes consume 36 mJ in the Kerberos protocol and 78 mJ for the Otway-Rees protocol per pair of SAFEMITS nodes. The Kerberos protocol is much better suited to DSN's that have low power synchronized clocks on their SAFEMITS nodes.

5.3.1.3 KDC Database Update Cost

A limitation of the key exchange methods presented above is that each KDC in the system needs to maintain a database of valid SAFEMITS nodes, which includes the keys that the KDC shares with the SAFEMITS nodes. These shared keys are typically unique to each KDC or KTC in the DSN as well, i.e. for each combination of SAFEMITS node and KDC a different key is used, so that the system can tolerate the compromise of a KDC an still continue to operate.

Before a SAFEMITS node is deployed, the shared key for every KDC that it will need to use throughout its lifetime must be installed on the SAFEMITS node. Likewise the KDC's database needs entries for SAFEMITS nodes that will be deployed in the future prior to the KDC's deployment or the database will need to be remotely updated by the Mission Authority after the

KDC is deployed on the battlefield. Such an update mechanism could be provided, but the communication energy cost of such an approach makes its use unattractive.

We assume that each KDC in the SAFEMITS network needs to receive a 20-byte record in order to support a new SAFEMITS node. The total DSN wide communication energy cost of updating the KDC's with varying average distances (in hops) from the Mission Authority to the KDC and different number of KDC's receiving an update is shown in Table 17. Here each KDC receives a single update message adding 12 new SAFEMITS nodes to the DSN. Each message contains 2000 bits (1920 + 80 bits of header) of information and we assume that all update messages take different paths.

Number of KDCs	Energy Consumption per Update Message (J)			
	4 Hops	8 Hops	16 Hops	32 Hops
3	0.82	1.64	3.28	6.56
6	1.64	3.28	6.56	13.13
12	3.28	6.56	13.13	26.25
24	6.56	13.13	26.25	52.50
48	13.13	26.25	52.50	105.00
96	26.25	52.50	105.00	210.00

Table 17 - Update Message Energy Consumption Required to Add 12 Nodes

In Sections 5.3.2 and 5.3.3 below describe two symmetric key based arbitrated methods that do not require that each KDC maintain databases for each SAFEMITS node. The first method uses symmetric key (i.e. secret-key cryptosystem) based certificates [Davis90]. This approach places the responsibility for storing the shared key on the SAFEMITS nodes. In the second method the secret keys that are shared by SAFEMITS nodes and KDC are generated based on the identity of the SAFEMITS node and secret information shared by the KDC and the DSN administrator. The trusted server generates the shared key when needed. This approach offers lower communication energy cost than does the symmetric certificate method and is equally as flexible.

5.3.2 Symmetric Key Certificate-Based Keying

In a symmetric key based certificate method, modified for SAFEMITS network use, the key that is shared between a SAFEMITS node and a trusted server (a KDC or KTC) is only stored on the SAFEMITS node and *not* on the server. In addition to its copy of the shared key, the SAFEMITS node also has a copy of a certificate for that shared key that only the proper KDC and the Mission Authority for the DSN can decrypt. For example a certificate that contains a secret key that should be shared by SAFEMITS Node A and KTC_j would have the form:

$$E(K_{Tj}, K_{Aj} \,||\, ID_A \,||\, L)$$

Where ID_A, is Node A's identifier, L is the lifetime of this certificate, K_{Aj} is the key that the KTC and Node A will share, and K_{Tj} is a secret key that the Mission Authority and the KTC_j share.

This approach offers a significant flexibility advantage over the traditional trusted server based protocols in Section 5.3.1. The trusted server does not store the shared keys, such as K_{Aj}, and does not have to protect and maintain a database of such keys. (See the 5.3.1.3 section for information on the energy costs of updating such a database after the KDC has been fielded.) However, it is up to the SAFEMITS nodes that are participating in certificate-based key establishment protocol to provide the necessary certificates to the appropriate trusted server and the energy cost of sending the certificates to the KTCs are significant.

The following protocol is a slightly modified version of a symmetric key certificate based key establishment protocol by Davis et. al [Davis90]. In this protocol, unlike the protocols in Section 5.3.1, one of the SAFEMITS nodes creates the key that it will share with its peer SAFEMITS node. This node has the responsibility of making sure that the shared key is not reused. Other symmetric key certificate based key establishment protocols assign key generation responsibility to the trusted server [Davis90].

The protocol

The protocol has four rounds. Each SAFEMITS, Node A, has a secret-key certificate $cert_{Aj} = E(K_{Tj},, K_{Aj} \| ID_A \| L)$ as explained above. The protocol initiator, Node A, generates two nonces M and N_A and encrypts these values; the identities of itself and its peer, Node B; and the session key Key_{pair}; using the secret key that it shares with the local key translation center KTC_j. In this protocol, the session key Key_{pair} is chosen by an ordinary SAFEMITS node rather than by the trusted server. Node A must properly generates the shared key to assure its freshness.

The encrypted value, Node A's symmetric certificate for KTC_j, the identities of all three participants and the nonce M are sent by Node A to B.

Round 1

Node A → Node B: $ID_B \| ID_A \| KDC_j \| M \| E(K_{Tj},, K_{Aj} \| ID_A \| L) \| E(K_{Aj}, Key_{pair} \| N_A \| M \| ID_A \| ID_B)$

Node B receives the message, optionally generates a nonce N_B, and encrypts the identities of the SAFEMITS nodes, N_B, and M, using K_{Bj}. Node B then takes the message it received, adds its own certificate $E(K_{Tj},, K_{Bj} \| ID_B \| L)$ for KTC_j, the optional encrypted value (which provides additional authentication) if necessary, and sends the result to KTC_j, as the Round 2 message.

Round 2

Node B → Node KTC_j: $KDC_j \| ID_A \| ID_B \| M \| E(K_{Tj},, K_{Aj} \| ID_A \| L) \| E(K_{Aj}, Key_{pair} \| N_A \| M, ID_A \| ID_B) \| E(K_{Tj},, K_{Bj} \| ID_B \| L) [\| E(K_{Bj}, N_B \| M \| ID_A \| ID_B)]$

KTC_j decrypts the certificates using K_{Tj} and uses the key, K_{Aj}, obtained from certificate $cert_{Aj}$, to decrypt $E(K_{Aj}, Key_{pair}, N_A \| M \| ID_A \| ID_B)$ and optionally uses K_{Bj} obtained from certificate $cert_{Bj}$, to decrypt $E(K_{Bj}, N_B \| M \| ID_A \| ID_B)$. KTC_j then performs the following actions depending on the variant of the protocol in use:

- **If the optional authentication is not done:** The KTC checks that the SAFEMITS node identity in $cert_{Bj}$ matches the second SAFEMITS node identity in $E(K_{Aj}, Key_{pair}, N_A \| M \| ID_A \| ID_B)$ so that the key K_{Bj} is the proper key for the SAFEMITS node that Node A wishes to establish a shared key with. The KTC also checks that M is used consistently throughout the message.
- **If the optional authentication is done:** The KTC performs the above checks and also checks that the SAFEMITS node identity in $cert_{Bj}$ matches the second SAFEMITS node identity in $E(K_{Aj}, Key_{pair}, N_A \| M \| ID_A \| ID_B)$ as in the above. In addition the KTC checks that Node B generated $E(K_{Bj}, N_B \| M \| ID_A \| ID_B)$.

The KTC encrypts the shared key and the nonces provided in the Round 2 message under the key it shares with Node B and sends the result with the necessary identifiers back to Node B as the Round 3 message.

Round 3

Node KDC$_j$ → Node B: $ID_B \,||\, ID_A \,||\, E(K_{Bj}\,,\, Key_{pair} \,||\, M \,||\, N_A\, [\,||\, N_B\,]\,)$

Node B decrypts the message and check that the nonce M is correct, or optionally that nonce N_B is correct. Node B then optionally encrypts the nonce N_A provided by Node A with the shared key generated by Node A and sends the result back to Node A. An alternative is to skip Round 4 and rely on the future use of the shared key to convince Node A that Node B knows the shared key (key confirmation).

Round 4

Node B → Node A: $[\, ID_A \,||\, ID_B \,||\, E(Key_{pair},\, N_A\,)]$

Node A decrypts the message and checks that the nonce N_A is correct, confirming that Node B knows the shared key.

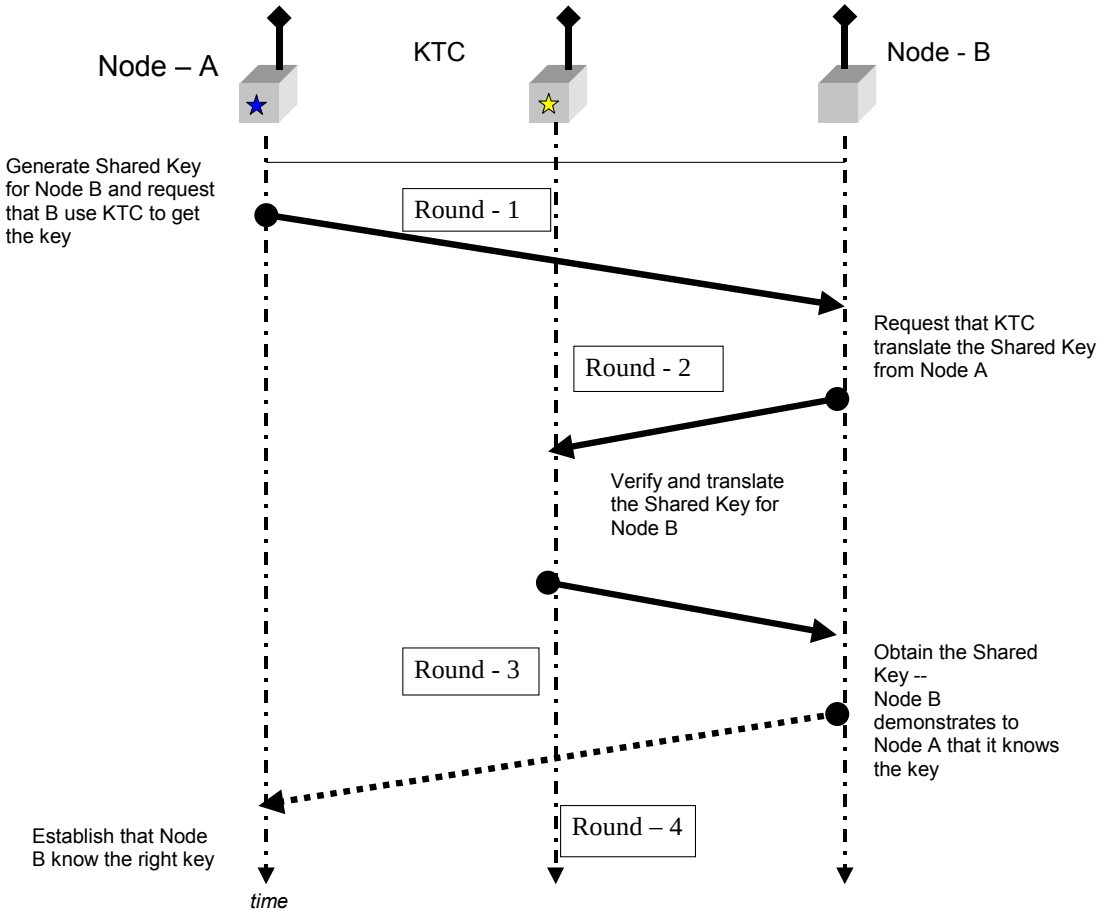

Figure 11 - Symmetric Certificate Protocol (modified for DSN use)

Analysis

The simpler version of the protocol provides one-way entity authentication, and a key freshness guarantee. The use of the optional message components and the fourth message provides mutual entity authentication, key confirmation and a key freshness guarantee. SAFEMITS Node A performs one secret-key encryption and optionally one secret-key decryption. SAFEMITS Node B performs two secret-key decryptions, one secret-key encryption and optionally one extra encryption.

Processor	Energy Consumption (mJ)			
	Node A	Node B	KDC	Node A + Node B
MIPS R4000	15.2	42	25	57
SA-1110 "StrongARM"	15.2	42	25	57
Z-180	16.4	44	27	60
MC68328 "DragonBall"	15.3	42	25	57
MCF5204 "ColdFire"	15.4	42	25	58
MMC2001 "M-Core"	15.2	42	25	57
ARC-3	15.2	42	25	57

Table 18 - Energy Consumption per Node for Symmetric Key Certificate Protocol

Under these assumptions:

- The node ID and KDC ID sizes are 64 bits.
- The symmetric keys are 128 bits.
- All nonces are 64 bits.
- The ID sizes are 32 bits.
- The lifetime field size is 64 bits.
- The Round 1 message is 640 bits.
- The Round 2 message is 1184 bits.
- The Round 3 message is 384 bits.
- The Round 4 message is 128 bits.

The total energy consumed by this protocol, when the three participants are neighbors, ranges from 81 mJ to 87 mJ. Almost all of the energy consumed by this protocol comes from the communications, not the computations. Only the Z-180 processor consumes a noticeable amount of computational energy when compared to the communications energy (approximately 7%). There is little variation in combined (computational and communication) energy consumption with processor type as is typical of all the secret-key based protocols we have examined to date. It is also worth noting that the energy consumed by the KDC is low per run of this protocol. This is also typical of the pairwise symmetric key based protocols.

In the Kerberos protocol the total energy consumed by the ordinary nodes ranges from 52 mJ to 55 mJ and the KDC nodes consumes between 20 and 21 mJ depending on processor type. These energy consumption numbers are significantly better than this symmetric certificate based protocol. However, in the Kerberos protocol the key distribution centers store all of the keys that they share with the SAFEMITS nodes so in order to compare the two protocols this update cost must be considered.

Since the Symmetric Key Certificate protocol uses a trusted server we need to consider how the energy cost of the protocol increases with increased distance (number of hops and hop distance) between the participants. In the next table we display estimated communication energy cost to the DSN of this protocol (total cost minus the KDC's cost) for a number of different hop counts between Node B and the KDC, nodes A and B are assumed to have no intermediate nodes. The total energy cost of the protocol to DSN (not including the KDC's cost) is essentially the same as the communication energy cost.

Number of Hops	Communications Energy Consumption (mJ)
1	52
2	88
4	130
8	220
16	390

Table 19 - Symmetric Key Certificate Protocol Multi-hop Communication Energy Cost

This protocol can be modified that the initiator does not specify the identity of its peer. This approach, in which the trusted server acts as a KDC rather than a KTC, allows the initiator to broadcast the first message of the protocol to a set of peer SAFEMITS nodes. This approach has slightly lower energy consumption and latency and is better suited for establishing shared keys between the initiator node and multiple local nodes than the protocol presented above.

This protocol has a drawback, which impacts its use in SAFEMITS network. Since each SAFEMITS node needs a different certificate for each KTC that it uses, to provide key granularity, the SAFEMITS nodes must be pre-configured with all of the certificates that each node will need prior to their deployment. Updating a large deployed SAFEMITS network to support a new KTC has prohibitive communication cost. Therefore, the Mission Authority must plan for the future and store extra certificates into the SAFEMITS nodes. As new KTC nodes are deployed the Mission Authority can assign each unused identity KTC_j. for which the currently deploy SAFEMITS nodes have corresponding certificates. This approach requires a great deal of computation on the part of the Mission Authority and is inadequate if the Mission Authority needs to join two already deployed SAFEMITS networks.

5.3.3 Identity-Based Symmetric Keying

An alternative key establishment technique that we have developed during this project is presented here. This approach, which we call Identity-Based Symmetric Keying (IBSK),[23] results in protocols that have lower energy costs than do symmetric key based certificate protocols and compare favorably (w.r.t. energy consumption) with traditional secret-key protocols such as Kerberos even when not considering the cost of doing KDC updates. IBSK based protocols, like the symmetric key certificate protocols, doe not require that the trusted servers maintain a database of keys that they share with the ordinary SAFEMITS nodes. Also like the symmetric key certificate protocol

[23] In IBSK, the key shared between a SAFEMITS and a super node is generated by inputting the identities of the SAFEMITS and super nodes to a key generation function. The notion of deriving a cryptographic key from the identity of a protocol principal is certainly not new. This identity based public key generation has been studied extensively, see [Schneier96] for a survey, and identity based symmetric keys have been used for authentication in [TMN] and in other protocols. The protocol presented in this section uses identity based symmetric keys for both confidentiality and authentication, and the authors believe that this protocol has not appeared before in the literature.

presented above IBSK protocols can use different keys for each KDC SAFEMITS node pair and use different shared keys between each KDC and the Mission Authority. This provides strong compromise protection.

In a DSN the KDC's or KTC's in an IBSK protocol do not store the keys that they share with the SAFEMITS nodes. Instead each trusted server uses a key that it shares with the Mission Authority and the identity of the SAFEMITS node to generate the shared key for that SAFEMITS node. The SAFEMITS node is loaded with the shared key prior to deployment. For example, KDC_j generates the key, K_{Aj}, that it shares with Node A by performing the following operation.

$$K_{Aj} = E(K_{Tj}, H(ID_A \| KDC_j))$$

Additional information such as an expiration time can be included in the input to the hash function. The energy advantage of IBSK protocols vs. symmetric certificate based protocol is that the messages are typically shorter; certificates are expensive to transmit and receive.

Protocol:

Node B generates nonce N_B and sends the following message to Node A requesting that it contact KDC_j to set up a shared key.

Round 1

Node B → Node A: $ID_A \| ID_B \| KDC_j \| N_B$

Node A receives the Round 1 message and generates the request message to send to KDC_j, where $M(K_{Aj}, KDC_j \| ID_A \| ID_B \| N_A \| N_B)$ is represent the use a message authentication code such as SHA-1. The first argument, K_{Aj}, is the key and the second is the data.

Round 2

Node A → Node KDC_j: $KDC_j \| ID_A \| ID_B \| N_A \| N_B \| M(K_{Aj}, KDC_j \| ID_A \| ID_B \| N_A \| N_B)$

KDC_j uses the identifier of Node A and the secret-key that it shares with the Mission Authority to generate K_{Aj}. The KDC then uses K_{Aj} to verify the MAC and if successful (and both Node A and Node B are still valid nodes) it generates the shared key Key_{pair} for nodes A and B. This shared key can be generates by encrypting the concatenation of the node identifiers and the nonces, i.e. $K_{pair} = E(K_{KDC\text{-}j}, ID_A \| ID_B \| R_A \| R_B)$, where $K_{KDC\text{-}j}$ is a secret key known only to KDC_j. KDC_j then encrypts the shared key and some verification information using K_{Aj} and K_{Bj}. The concatenation of these two encryptions plus the identifiers of the SAFEMITS nodes is the Round 3 message.

Round 3

Node KDC_j → Node A: $ID_A \| ID_B \| E(K_{Aj}, H(ID_A \| ID_B \| N_A \| N_B) \| Key_{pair}) \| E(K_{Bj}, H(ID_A \| ID_B \| N_A \| N_B) \| Key_{pair})$

Node A decrypts $E(K_{Aj}, H(ID_A \| ID_B \| N_A \| N_B) \| Key_{pair})$, recovers Key_{pair}, and verifies $H(ID_A \| ID_B \| N_A \| N_B)$. If the verification succeeds Node A replaces $E(K_{Aj}, H(ID_A \| ID_B \| N_A \| N_B) \| Key_{pair})$ with N_A, generates $E(Key_{pair}, N_A \| N_B)$ to prove to Node B that Node A received the shared key, and sends the resulting message to Node B.

Round 4

Node A → Node B: $ID_B \| ID_A \| N_A \| E(K_{Bj}, H(ID_A \| ID_B \| N_A \| N_B) \| Key_{pair}) \| E(Key_{pair}, N_A \| N_B)$

Node B decrypts $(K_{Bj}, H(ID_A \| ID_B \| N_A \| N_B) \| Key_{pair})$, recovers Key_{pair}, and verifies $H(ID_A \| ID_B \| N_A \| N_B)$. Node B then decrypts $E(Key_{pair}, N_A \| N_B)$ and verifiers its contents. If the verification succeeds Node B generates a shared key confirmation message and sends it to Node A.

Round 5

Node B → Node A: $ID_A \| ID_B \| E(Key_{pair}, N_B \| N_A)$

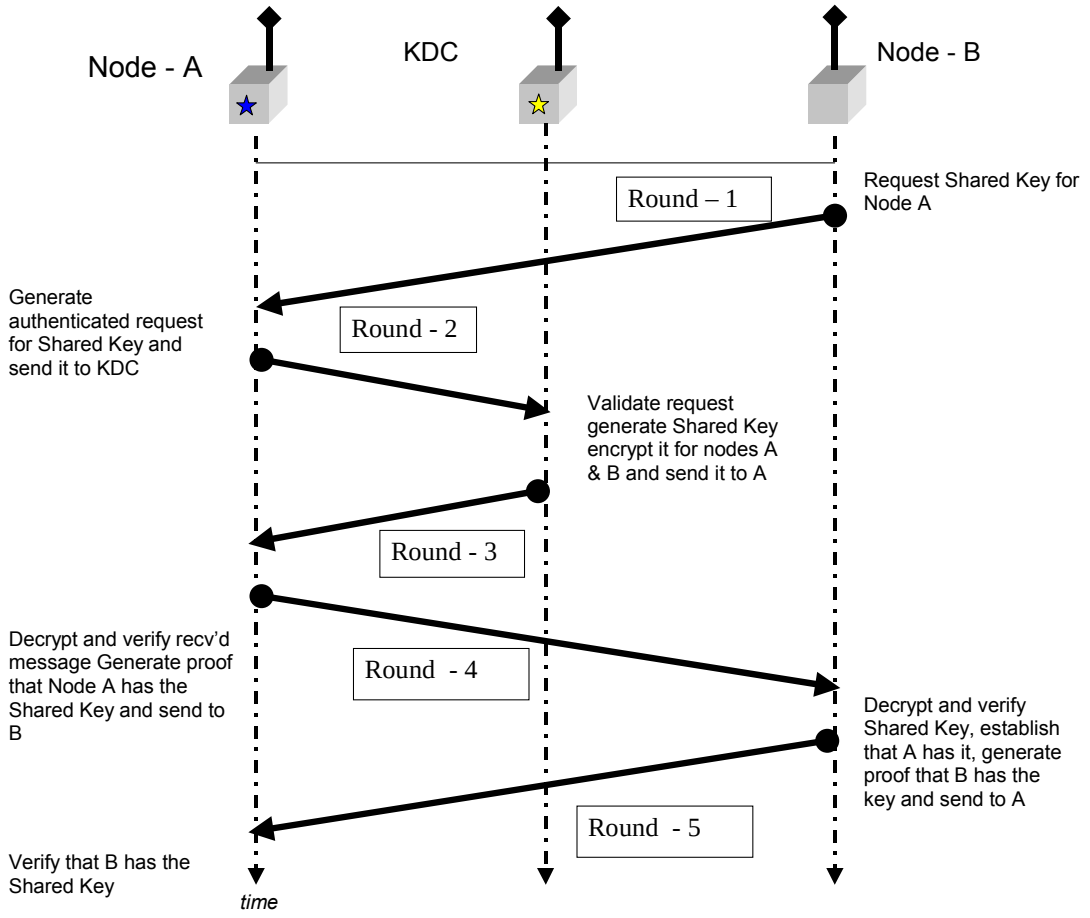

Figure 12 - Identity Based Symmetric Keying Protocol

Analysis

The protocol provides mutual entity authentication, key confirmation and a key freshness guarantee. Each SAFEMITS node (nodes A and B) performs two secret-key decryptions and one secret-key encryption. Under these assumptions:

- The node ID and KDC ID sizes are 64 bits.
- The symmetric keys are 128 bits.
- All nonces are 64 bits.
- The Round 1 message is 768 bits.
- The Round 2 message is 1280 bits.

- The Round 3 message is 960 bits.
- The Round 4 message is 896 bits.

As shown in Table 20, the computational energy cost of this method is relatively low, and is less than 5% of the total communication cost (71 mJ) of the SAFEMITS nodes for the inefficient Z-180 processor. The total energy cost of the protocol when the participants are neighbors ranges from 94 mJ to 100 mJ.

Processor	Computational Energy Costs (mJ)	
	Node A + Node B	KDC
MIPS R4000	0.0139	0.0114
SA-1110 "StrongARM"	0.0262	0.021
Z-180	2.9	2.4
MC68328 "DragonBall"	0.157	0.129
MCF5204 "ColdFire"	0.46	0.38
MMC2001 "M-Core"	0.036	0.029
ARC-3	0.001	0.0008

Table 20 - Identity-Based Symmetric Keying Computational Energy Consumption

The IBSK protocol costs the ordinary SAFEMITS nodes on average only 84% of the energy of the symmetric certificate protocol presented in Section 5.3.2 and is approximately 92% of the energy of the Kerberos protocol. The KDCs of the IBSK protocol, like the symmetric certificate protocol but unlike the Kerberos or Otway-Rees protocols, do not need to store the keys that they share with the SAFEMITS nodes.

The IBSK and Symmetric Certificate Protocols do not need to be updated when new nodes are deployed. In the Symmetric Certificate Protocol the KDC learns about SAFEMITS nodes via the certificates that it receives during the protocol, in IBSK the KDC only needs an identifier for a SAFEMITS node in order to establish a shared key with the SAFEMITS. In all of the protocols in Section 5.3.1 when a new KDC is deployed those SAFEMITS nodes that will make use it must have been deployed with the necessary key for that KDC or they must be updated in the field. The cost of updating the SAFEMITS nodes grows linearly with the number of SAFEMITS nodes and linearly with the average "distance" between the SAFEMITS nodes and the closest gateway node. The Mission Authority must generate and transmit (and the DSN forward) a unique message for each SAFEMITS node (for some region of the DSN).

In IBSK the Mission Authority must plan for the future and store extra K_{Aj} keys in the SAFEMITS nodes. As new KDC nodes are deployed the Mission Authority can assign each unused identity KDC_j. for which the currently deployed SAFEMITS nodes have corresponding certificates. This approach requires a great deal of computation on the part of the Mission Authority and is inadequate if the Mission Authority needs to join two already deployed SAFEMITS networks.

Applying Identity Based Keying to the Kerberos and Otway-Rees Protocols

In the Kerberos and Otway-Rees protocols the key shared by each SAFEMITS node and super node *could* be generated using an IBSK key generation equation such as $K_{Aj} = E(K_{Tj}, H(ID_A \,||\, KDC_j))$. This simple extension of these protocols does not affect the messages that make up each protocol but it eliminates the energy cost of updating the super node databases, see Table 17, when new SAFEMITS nodes are deployed and reduces storage costs on the super nodes. Eliminating the database updates saves considerable energy, for example the estimated communication energy cost of updating 24 super nodes that are each an average of 16 hops away from a gateway nodes about the deployment of 12 SAFEMITS nodes each is 26.25 Joules or approx. 2.19 Joules per SAFEMITS node. Compare this value with the cost of performing the Kerberos protocol where the SAFEMITS

nodes are also about 16 hops away from the super node, and we see that updating the databases is 5 times more costly that running the protocol itself in this instance.

5.3.4 Arbitrated Group Keying Protocols

This document examines key establishment techniques with the goal of providing security services for DSN system operations such as routing. In this context we are primarily focused on pairwise and small group key establishment. Furthermore since the DSN nodes are assumed not to be mobile nodes, as is the case in general ad-hoc networks, the membership of the groups is relatively static.

5.3.4.1 Small Group Extensions of Arbitrated Pairwise Protocols

The pairwise key establishment protocols of Sections 5.3.1, 5.3.2 and 5.3.3 can be extended to support key establishment for small groups in a relatively straightforward manner once the potential membership of the group has been established. The Kerberos protocol can be extended to the small group setting by a straightforward modification of the standard protocol. When the protocol begins, the *initiator* node of the group and the potential membership of the group have already been determined.

The protocol

Each SAFEMITS node *i* has a secret key that it shares with a KDC (for simplicity sake we will assume that there is one KDC in the system, KDC_j. ID_A is the unique identifier of node A, likewise ID_B for node B. N_A is a random nonce that is used (with sufficiently high probability) no more than once for the same purpose [Menezes, p397]. Node A has the role of the group initiator and the potential membership of the group is {Node A, Node B, Node C, Node D, Node E, Node F}.

Round 1

Node A → Node KDC_j: $KDC_j \| ID_A \| ID_B \| ID_C \| ID_D \| ID_E \| ID_F \| N_A$

The initiator, Node A, sends the identity of itself and the other proposed group members, who we assume Node A already knows, to KDC_j. The KDC looks up Node A through Node F in its database, verifies that they are valid supported nodes, and fetches their corresponding shared keys K_{Aj} through K_{Fj}.

Define $ticket_B$ as $E(K_{Bj}, K_{group} \| ID_A \| ID_C \| ID_D \| ID_E \| ID_F \| L)$ where K_{group} is the group key that will be shared by the subset of {Node A … Node F} approved by the KDC. L is the lifetime of the key. The ticket, $ticket_B$ can only be decrypted by Node B. The other tickets are defined analogously. The KDC may reject some of the member of the group or disallow the group entirely. If Node A is not a member of the group then we will assume that the group is rejected, other strategies are possible by extending the protocol. Node A obtains the group key by decrypting $E(K_{Aj}, K_{group} \| ID_B \| ID_C \| ID_D \| ID_E \| ID_F \| N_A \| L)$. The KDC sends the following message back to Node A.

Round 2

Node KDC_j → Node A: $ID_A \| KDC_j \| ticket_B \| ticket_C \| ticket_D \| ticket_E \| ticket_F \| N_A \| E(K_{Aj}, K_{group} \| ID_B \| ID_C \| ID_D \| ID_E \| ID_F \| N_A \| L)$

Node A decrypts $E(K_{Aj}, K_{group} \| ID_B \| \ldots \| ID_F \| N_A \| L)$ and verifies that ID_B through ID_F (or whatever subset of this list the KDC approved) and N_A match the message what Node A sent to the KDC in Round 1. Then Node A takes a fresh timestamp T_A, creates the Round 3 messages and sends them to the other nodes that make up the group.

Round 3

Node A → Node X: $ID_X \;||\; ID_A \;||\; ticket_B \;||\; E(K_{group,}\; ID_X \;||\; [other\ group\ member's\ IDs\ ||]\; T_A)$

When it receives its message, each Node X, decrypts $ticket_X$ and obtains K_{pair}, and uses it to decrypt $E(K_{group,}\; ID_X \;||\; T_A \;||\; N_{AX})$. The nodes can verify the identifiers obtained from both encrypted values, that the timestamp T_A is still valid and that Node B's local time is within the lifetime L. Each node encrypts the timestamp and nonce provided by Node A with the shared key message and sends its Round 4 message to Node A.

Round 4

Node X → Node A: $ID_A \;||\; ID_X \;||\; E(K_{group,}\; T_A \;||\; ID_X)$

Node A decrypts the messages and verifies that the timestamp and nonce match what it sent Node X in Round 3. This message allows Node A to determine that Node X received the Round 3 message and successfully decrypted the group key. Notice that any of the other group members can successfully claim to Node A that another group member has received the group key.

Analysis

This extension of the Kerberos protocol provides *mutual entity authentication*, *limited* key confirmation and a key freshness guarantee. Each SAFEMITS node performs two secret-key decryptions and one secret-key encryption. Under these assumptions:

- The size of the proposed group is 6 and the KDC approves of the entire group
- The node ID and KDC ID sizes are 64 bits.
- The symmetric keys are 128 bits.
- All nonces are 64 bits.
- The timestamps and validity periods (lifetimes) are 64 bits.
- The Round 1 message is 512 bits.
- The Round 2 message is 4032 bits.
- The Round 3 messages are 896 bits, assuming no extra IDs are sent.
- The Round 4 messages are 384 bits, since including the extra ID does not increase length of the ciphertext.

The communication energy costs are: 188 mJ for Node A, 21 mJ for each of the other group members, 91 mJ for the KDC. As is shown in Table 21, the computational energy cost of this method is relatively low. The total energy cost of the protocol is essentially the same as the communication cost for all of the processors except for the Z-180, where the total cost ranges from 5% to 10% more than the communication cost (single hop scenario). The cost of this protocol can be reduced by replacing the ID's within the tickets with a hash of the IDs and a bitmap of which of the proposed group members have been accepted, a saving of about 190 bits per ticket.

Processor	Computational Energy Consumption (mJ)		
	Node A	Nodes B … F	KDC
MIPS R4000	0.023	0.015	0.063
SA-1110 "StrongARM"	0.043	0.028	0.12
Z-180	4.8	3.1	13
MC68328 "DragonBall"	0.26	0.17	1.0
MCF5204 "ColdFire"	0.76	0.49	2.1
MMC2001 "M-Core"	0.059	0.038	0.16
ARC-3	1.6×10^{-3}	1.0×10^{-3}	4.4×10^{-3}

Table 21 – Group Kerberos Protocol Computational Energy Consumption (group size = 6)

An important consideration with protocols that use trusted servers is the total energy cost of the protocol (minus the cost to the energy rich trusted server). The energy cost of the protocol increases with the increase distance (number of hops and hop distance) between the participants. The earlier communication energy costs were based on the assumption that Node A was one hop away from the other nodes in the group and we did not include the energy cost imposed on any intermediate nodes between Node A and the KDC. In the next table we display estimated communication energy cost to the DSN of this protocol (minus the KDC cost) for a number of different hop counts between Node A and the KDC. The total energy cost is essentially the same as the communication energy cost.

Number of Hops	Communications Energy Consumption (mJ)
1	293
2	408
4	636
8	1090
16	2010

Table 22 – Group Kerberos Protocol Multi-hop Communication Energy Consumption (group size = 6)

5.3.4.2 Logical Key Hierarchy

Wallner et. al. propose a hierarchical keying method [Walner98] called Logical Key Hierarchy (LKH) that facilitates the distribution of a common group key to a large number of participants. In this method, a central trusted third party (TTP) initiates the creation of a logical key hierarchy constructed of *key-encryption-keys (KEKs)* with a single group key, GTEK, at the root of the tree (see Figure). The leaves of the tree correspond to unique KEKs shared between each participant and the TTP. Each tier above a participant corresponds to a different KEK. Every participant has knowledge of only the KEKs in a direct path from their leaf node to the root. In the event of a compromise, the TTP can use the hierarchical tree of KEKs to efficiently rekey the compromised portions of the tree, including the GTEK.

The Protocol

LKH is not specifically a protocol but rather a key management technique for efficiently keying a large group of participants. Within a SAFEMITS network, the hierarchical technique can be applied in support of other keying protocol such as *pre-deployed keying* (see Section 5.2) to provide a mechanism to maintain freshness of the shared deployed cryptographic keys.

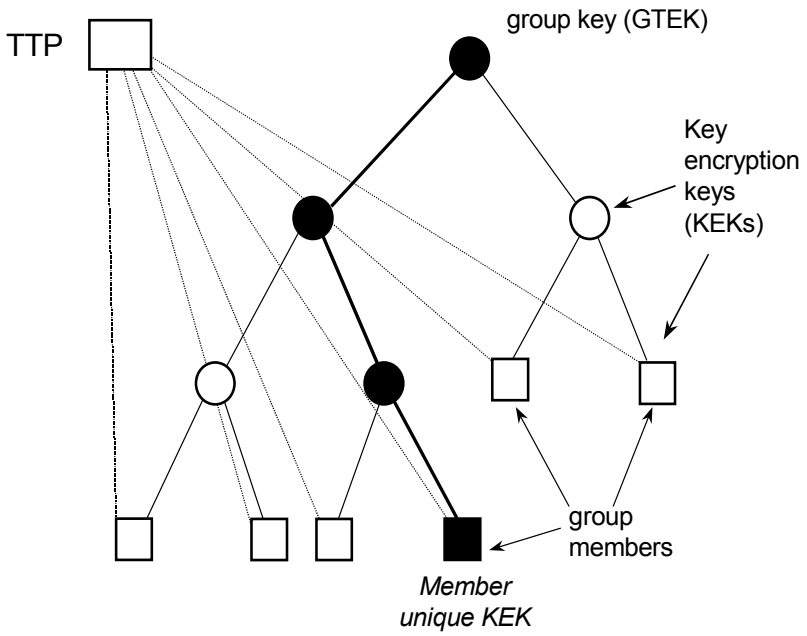

Figure 13 - Logical Key Hierarchy (LKH)

Analysis

Because of its hierarchical structure, LKH can perform rekey operations with logarithmic efficiency. The number of messages required to rekey the tree is *(k-1)d* for a *k-ary* tree of depth *d*. The creation of the initial tree requires the exchange of secret parameters between each participant and the TTP. This peer-to-peer communication is probably best performed prior to deployment of a SAFEMITS network to minimize the linear cost of creating the tree with its group members. Otherwise, the TTP would have the energy expense of contacting each SAFEMITS node individually to establish the unique shared key. The transmission cost to at the TTP to initialize the group is *2nK+d* for *n* participants, a tree of depth *d*, and a key size of *K* [McGrew]. Thus, for a 100 SAFEMITS nodes organized into a logical binary-tree of depth 7, requires approximately 25.6 kbits to be transmitted by the TTP to initialize the group using 128-bit keys at a cost of 538 mJ.[24]

In the event of a compromise or to maintain freshness of keys, LKH can be applied to efficiently rekey portions of a DSN sharing a common group key. The computational energy costs to the network are centralized at the TTP who must complete the logical tree by encrypting and distributing the portions of the tree to affected SAFEMITS nodes. The total broadcast size of the rekey message is approximately *2dK+d* [McGrew]. The resulting transmission energy consumption at the TTP is approximately 38 mJ to evict a single member from the tree. The rekey computation costs associated with each node is negligible, consisting of decrypting the wrapped or encrypted keys of the tree. A more significant cost is the cost to transmit the rekey messages.

The method requires each participant store *d+1* keys. The TTP must store all the keys in the tree and perform the key encryption functions required for distribution of new GTEKs when required

[24] For the SAFEMITSia WINS NG RF subsystem transmitting at 10 kbps with 10mW of power.

due to compromise or cryptoperiod expiration. In this centralized role, the TTP is vulnerable to denial of service attacks as a central point of failure in the method.

The centralized nature of this hierarchical keying approach is most efficient in a multicast communications environment. However, in a multi-hop SAFEMITS network, the energy efficiency benefits of LKH are vastly outweighed by the communications cost of routing keying messages over multiple hops.

5.3.4.3 One-way Function Tree

Balenson, McGrew, and Sherman [McGrew97] present a hierarchical group keying method that establishes a common group key for participants by "pulling" the key from the root using *one-way functions* rather than "pushing" it from the root using key-encryption-keys as in LKH.

The Protocol

A randomly chosen key is assigned to each member that resides at leaves in a binary tree; the root is the group key. Each node in the tree is associated with two cryptographic keys: a node key k_x and a blinded node key k'_x computed from the node key using a one-way function g where $k'_x = g(k_x)$. The one-way blinding function can be a cryptographic hash function such as SHA-1, MD5, or even a simple XOR requiring significantly less energy than public key operation.

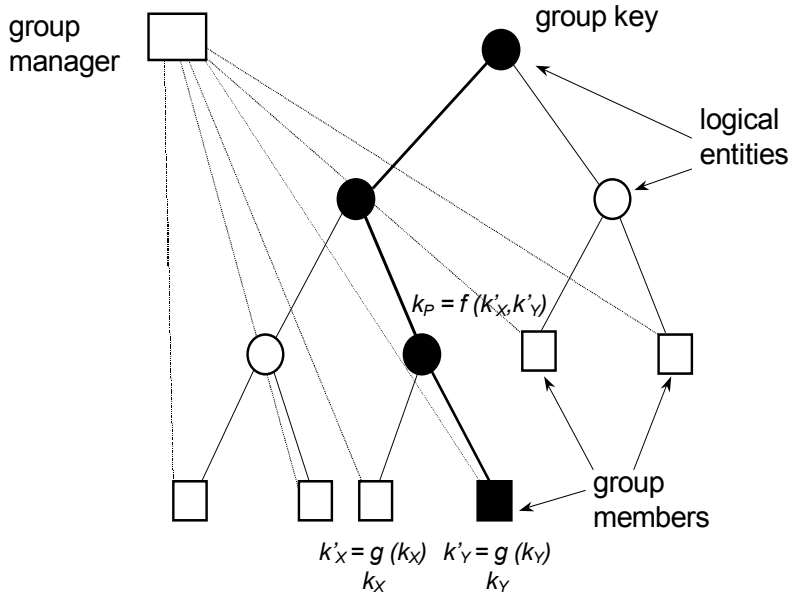

Figure 14 - A One-Way Function Tree (OFT)

The TTP maintains the entire one-way function tree. Each member in the tree knows the unblinded node keys on a direct path from its node to the root, and the blinded node keys that are siblings along this same path. The member knows no other keys in the tree. The number of keys stored by group members and the number of keys required to be broadcast when new members are added or evicted is logarithmic providing a scalable linear rekey performance to support compromise or freshness requirements. Unlike LKH, the energy costs required to generate the group key is distributed throughout the group. Interior node keys are defined by the rule:

$$k_p = f(g(k_x), g(k_y)),$$

where *x* and *y* denote the left and right child of the node *p*, respectively. With the proper blinded and unblinded node keys, each group member can construct the root group key.

Analysis

As in LKH, establishment of the tree is probably best performed prior to deployment of a SAFEMITS network to minimize the linear cost of creating the tree with its group members. Otherwise, the TTP encounters the energy expense of contacting each SAFEMITS node individually to establish the unique shared key. This cost is identical to LKH.

The benefit of OFT's hashing functions is evident its rekey performance. An OFT rekey broadcast message size is on the order of $dK+d$ bits for a tree of depth d and key size K [McGrew]. Thus, a 128-bit key and a logical tree of depth 7 requires roughly half as much energy as LKH, only 19 mJ, to evict a single node from the tree.[25]

As with the LKH protocol, the centralized nature of OFT is most efficient in a multicast communications environment. However, in a multi-hop SAFEMITS network, the energy efficiency benefits of OFT keying are vastly outweighed by the communications cost of routing keying messages over multiple hops.

5.3.5 Energy Consumption Shifting Key Establishment Protocols

Distributed SAFEMITS networks will contain a small numbers of nodes with additional communications capabilities and available energy. These energy-endowed *super* nodes include gateway nodes with long-haul communications capabilities, vehicular-mounted communications nodes, and soldier radio nodes. Nearby generic SAFEMITS nodes can shift their key management computational energy burden to these super nodes by using certain key establishment protocols that leverage the asymmetric computational characteristics of some public key cryptographic algorithms, especially RSA, to minimize the computational energy costs of establishing a common key between two SAFEMITS nodes. In this section we will discuss arbitrated energy consumption shifting key establishment protocols with the energy-endowed nodes functioning as the arbitrator, in Section 5.3.5 we will discuss energy consumption shifting key establishment protocols that do not use trusted third parties.

Research into arbitrated energy consumption shifting keying protocols began with the work of Tatebayashi, Matsuzaki, and Newman [Tate89]. They proposed that mobile devices could establish pairwise shared keys by using public key cryptography and a trusted third party that assumes the majority of the computational cost of the protocol.

The mobile nodes use RSA to encrypt "keying material" with the public key of a trusted third party and send the results to the TTP. The TTP decrypts the messages and generates a secure message using a symmetric key algorithm that contains the session key that the mobile nodes should use (generated by one of them). This message is sent back to the other mobile node that decrypts the message to recover the key.

The original TMN protocol (KDP1) using the description from [Tate89]:

Node A generates a random value R_A in Z_N and encrypts R_A using the RSA public key of the trusted third party S and sends the result in a message to S.

[25] For the SAFEMITSia WINS NG RF subsystem transmitting at 10 kbps with 10mW of power.

Round 1

Node A → TTP S: $E(K_{Public\text{-}S}, R_A)$

The trusted third party S decrypts the message and recovers R_A. The trusted third party then calls Node B.

Round 2

TTP S → Node B: *"S calling B"*

When Node B receives this message it generates a random value R_B in Z_N and encrypts R_B using the RSA public key of the trusted third party S and sends the result in a message to S.

Round 3

Node B → TTP S: $E(K_{Public\text{-}S}, R_B)$

The trusted third party S decrypts the message and recovers R_B. The trusted third party then "encrypts R_B by a key-encryption key R_A and sends the result, $E(R_A, R_B)$, to Node A.

Round 4

TTP S → Node A: $E(R_A, R_B)$,

Node A decrypts $E(R_A, R_B)$, and uses the result R_B as the session key for Node A.

This protocol has a serious flaw that was first pointed out by Simmons [Tate89, see also Simmons94], the author's revision of the above protocol (called KDP2) was broken by Park et al. [Park94] who proposed a fix. Other authors have found different attacks on KPD1 and KDP2 and proposed other variants of the protocols, see [Lowe97].

5.3.5.1 The Rich Uncle Protocol(s)

In this section we propose a protocol that uses the same principles as the TNM protocol and it variants. This protocol, which we call Rich Uncle, differs from the TNM variations in a number of respects:

1. All of the mobile nodes can contribute to the value of the shared group key. This provides superior security.
2. All of the mobile nodes initially contact the TTP. In the TNM protocol the initiator mobile node contacts the TTP and the TTP sends a message in the clear to the other mobile node informing it that the initiator node is trying to establish a shared key. The TNN approach is better suited for cell-phones and other such applications while the Rich Uncle approach is better for DSNs.
3. Some of the TNM protocol variations use a secret shared by each mobile node and the trusted third part to authenticate the mobile node to the trusted third party in rounds 1 and 3 of the protocol. This approach greatly limits the flexibility of the protocol. The Rich Uncle protocol uses certificates signed by the DSN Authority that contain a "hash" of a secret known only to each mobile node rather than having each trusted third party in the system store a copy of these secrets. In Rich Uncle each mobile node authenticates itself to the trusted third party by presenting its certificate and its secret (encrypted using the public key of the TTP) to the trusted third party. This approach eliminates the need for each TTP to maintain a long-term database of the secrets of the SAFEMITS nodes, which weakens the security of the DSN and limits its flexibility.

The Protocol

As shown in Figure , the protocol begins by the super node sending an RSA-signed public key certificate (or potentially chain of certificates) to nearby nodes during the routing phase. Each SAFEMITS node verifies the certificate (or chain of certificates) to a public key root loaded during manufacturing or pre-deployment.

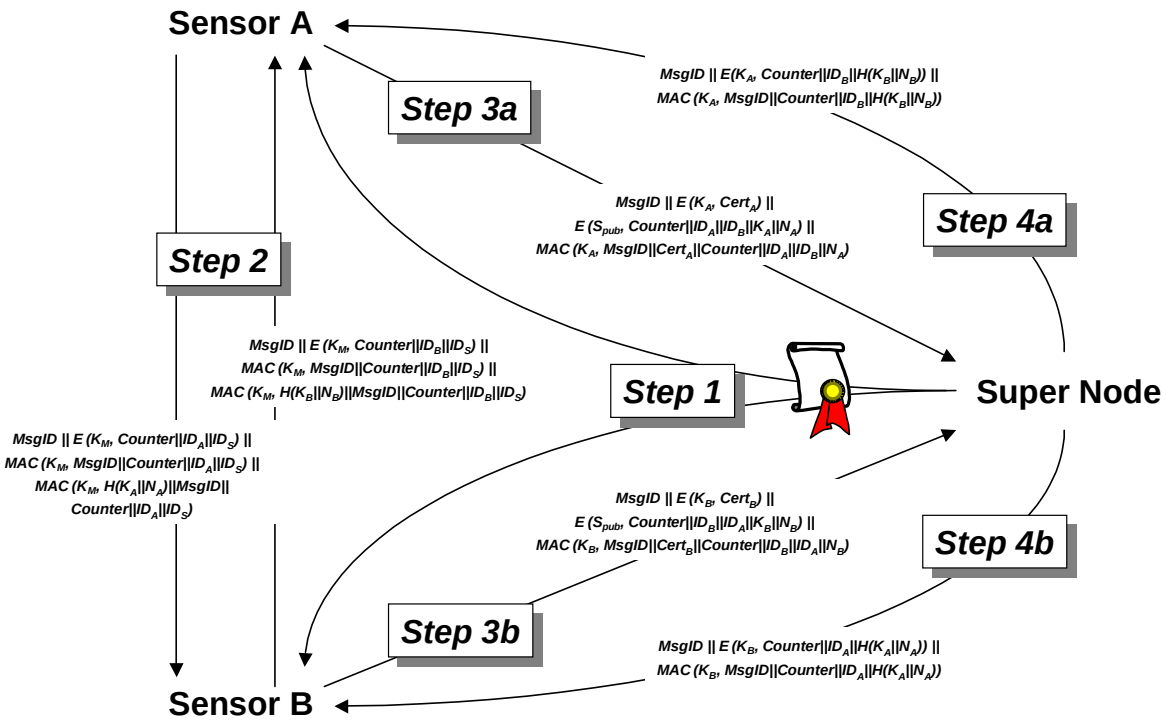

Figure 15 - Rich Uncle Keying Protocol

In step 2 shown in Figure , the nodes exchange messages of the form:

$$Message2 = MsgID \;||\; E(K_M, Counter \;||\; ID_i \;||\; ID_S) \;||$$
$$MAC(K_M, MsgID \;||\; Counter \;||\; ID_i \;||\; ID_S) \;||$$
$$MAC(K_M, H(K_i \;||\; N_i) \;||\; MsgID \;||\; Counter \;||\; ID_i \;||\; ID_S)$$

where,

MsgID is a 32-bit value used to identify this message as *Message2*, since the remainder of the message is encrypted,

Counter is a 32-bit counter value used to prevent replay attacks,

ID_i is the transmitting node's ID number,

ID_S is a super node ID number,

$MAC(K_M, \ldots)$ denotes use of a message authentication code, such as the HMAC-SHA-1-96 algorithm, using a pre-deployed network-wide mission key,

K_i is node i's symmetric key,

N_i is random nonce information generated by node i and used only for this key exchange,

$H(\ldots)$ denotes use of a hash function, such as SHA-1, and

|| denotes concatenation.

Exchanging *Counter* discourages replay attacks by an adversary that records and then re-transmits a node's legitimate transmission, and also obviates the need for an initialization vector by ensuring the uniqueness of the first plaintext encryption block. Receiving nodes must maintain previous counter values of *Message2* messages to ensure new counter values are used for each new exchange attempt. The purpose of exchange ID_i is for binding purposes later in the protocol. The purpose of exchanging ID_S is to synchronize the two SAFEMITS node participants on the chance that two or more super nodes are within one (or more) hops of either node. All message parameters in Messages 2 through 4, except *MsgID*, are encrypted for confidentiality protection of the exchanged key material, the exchanged integrity/authentication tags, and the identities of the exchanging parities. Exchanging the $MAC(K_M, MsgID \,||\, Counter \,||\, ID_i \,||\, ID_S)$ ensures legitimate receivers that the transmitting node knows the pre-deployed network-wide mission key and that the transmission is unique, further providing a modest level of denial of service protection. The $MAC(K_M, H(K_i \,||\, N_i) \,||\, MsgID \,||\, Counter \,||\, ID_i \,||\, ID_S)$ term is used later in the protocol to bind the key material received from the super node (i.e. $H(K_i \,||\, N_i)$) with the other information received from node i in *Message2*.

In step 3 of the Rich Uncle protocol, each participating node sends the super node its contribution to the shared key using the following message:

$$Message3 = MsgID \,||\, Cert_i \,||\, E(S_{pub}, Counter \,||\, ID_i \,||\, ID_j \,||\, K_i \,||\, N_i) \,||\, MAC(K_i, MsgID \,||\, Cert_i \,||\, Counter \,||\, ID_i \,||\, ID_j \,||\, N_i)$$

where,

MsgID, Counter, ID_i, K_i, and N_i are as described above,

ID_j is the ID number of the node that node i wishes to establish a key,

$Cert_i$ is a certificate that contains $H(K_i)$, and is signed by a mission authority trusted by the system root, and

S_{pub} is the super node's public key.

Step 3 provides the super node with a unique proof of the node's credentials, contribution to the shared key, the identity of the other node, and a key for protecting key material sent back to node i by the super node. The node's credentials are established with the super node by providing the K_i corresponding to the $H(K_i)$ contained in the verifiable $Cert_i$. Node i's contribution to the shared key is $K_i \,||\, N_i$. The MAC provides binding integrity between all message parameters. *Message4* is protected using the K_i provided in *Message3*.

In Step 4, the super node forwards each node's key material contribution to the other node to allow each node to compute the shared key. This last message is of the form:

$$Message4 = MsgID \,||\, E(K_i, Counter \,||\, ID_j \,||\, H(K_j \,||\, N_j)) \,||\, MAC(K_i, MsgID \,||\, Counter \,||\, ID_j \,||\, H(K_j \,||\, N_j))$$

where,

MsgID, Counter, ID_j, K_i, K_j, and N_j are as described above, except the subscript *j* denotes ID and key contribution from another node.

In addition to the other node's key material, the super node authenticates the provided key material to ensure the receiving node of the legitimacy of both the other SAFEMITS node and its key contribution. Authentication is provided through the use of a MAC based on the same unique symmetric key provided by the SAFEMITS node to the super node in *Message3*.

Energy Consumption

The goal of the Rich Uncle technique is to shift the computational burden of key establishment from the low-energy SAFEMITS nodes to energy-endowed super nodes. However, if computational energy consumption is not a large portion of the overall energy consumption, the Rich Uncle benefit is largely negated.

For instance, the WINS SAFEMITS nodes for the FY 2000 SAFEMITS experiment will use the very capable R4400 processor as the main processor, but will also use a less energy efficient RF subsystem for transmission and reception of key exchange information. As shown in Table 23, the energy costs of the Rich Uncle technique show the ratio of overall energy consumption between super and SAFEMITS node is less than a factor of two, even though the computational energy consumption ratio is over a factor of ten.

Step	Consumption Type	SAFEMITS Node Energy (mJ)	Super Node Energy (mJ)
Step 1	Communications	28.00	84.00
	Computation	0.81	0.00
	Step 1 Sub Total	28.81	84.00
Step 2	Communications	12.32	0.00
	Computation	0.00	0.00
	Step 2 Sub Total	12.32	0.00
Step 3	Communications	49.06	65.41
	Computation	0.81	16.72
	Step 3 Sub Total	49.87	82.13
Step 4	Communications	5.38	16.13
	Computation	0.00	0.00
	Step 4 Sub Total	5.38	16.13
Totals	Communications	94.75	165.54
	Computation	1.62	16.72
	Total of All Steps	96.37	182.26

Table 23 - Rich Uncle Energy Consumption for MIPS R4000 with WINS Communications

If the WINS RF subsystem is substituted with the more energy-efficient MuRF RF subsystem [Philsar00], the Rich Uncle energy consumption costs shown in Table 24 drop as expected. More interestingly, the ratio of energy consumption between the super and SAFEMITS nodes increases.

Step	Consumption Type	SAFEMITS Node Energy (mJ)	Super Node Energy (mJ)
Step 1	Communications	4.21	28.94
	Computation	0.81	0.00
	Step 1 Sub Total	5.02	15.20
Step 2	Communications	3.29	0.00
	Computation	0.00	0.00
	Step 2 Sub Total	3.29	0.00
Step 3	Communications	16.90	9.84
	Computation	0.81	16.72
	Step 3 Sub Total	17.71	26.56
Step 4	Communications	0.81	5.56
	Computation	0.00	0.00
	Step 4 Sub Total	0.81	5.56
Totals	Communications	25.21	44.34
	Computation	1.62	16.72
	Total of All Steps	26.84	61.06

Table 24 - Rich Uncle Energy Consumption for MIPS R4000 with MuRF Communications

Since SAFEMITS nodes must be inexpensive, it is possible future node configurations will use less capable processors such as the Motorola Dragonball processor that is deployed in millions of Palm Pilots. If the Palm Pilot's Dragonball processor is substituted for the MIPS R4000, the ratio of super to SAFEMITS node increases even more dramatically.

Step	Consumption Type	SAFEMITS Node Energy (mJ)	Super Node Energy (mJ)
Step 1	Communications	4.21	28.94
	Computation	42.16	0.00
	Step 1 Sub Total	46.37	28.94
Step 2	Communications	3.29	0.00
	Computation	0.00	0.00
	Step 2 Sub Total	3.29	0.00
Step 3	Communications	16.90	9.84
	Computation	42.16	840.25
	Step 3 Sub Total	59.06	850.09
Step 4	Communications	0.81	5.56
	Computation	0.00	0.00
	Step 4 Sub Total	0.81	5.56
Totals	Communications	25.21	44.34
	Computation	84.31	840.25
	Total of All Steps	109.52	884.59

Table 25 - Rich Uncle Energy Consumption for Dragonball with MuRF Communications

Table 26 compares four SAFEMITS node compositions, showing how the Rich Uncle method provides its greatest benefit when a less capable processor and energy-efficient RF subsystem are mated together.

Node Composition	SAFEMITS Node Energy Consumed (mJ)	Super Node Energy Consumed (mJ)	Energy Ratio, Super/SAFEMITS
R4000 Processor w/ WINS Communications	96.37	182.26	1.89
R4000 Processor w/ MuRF Communications	26.84	61.06	2.28
Dragonball Processor w/ WINS Communications	179.07	1005.79	5.62
Dragonball Processor w/ MuRF Communications	109.52	884.59	8.08

Table 26 - Relative Energy Consumption for Various Rich Uncle Scenarios

The Rich Uncle energy consumption costs are based on the following assumptions:

- The public key certificate infrastructure is two levels.
- The RSA modulus size is 1024 bits.
- The RSA public exponent is 65537.
- The message ID size is 32 bits.
- The node ID size is 32 bits.
- All symmetric keys and nonces are 128 bits.
- The counter size is 32 bits.
- The authentication tag size is 96 bits.
- The super node certificate size is 2000 bits.
- The SAFEMITS node certificate size is 1184 bits.
- The computational energy costs of AES and SHA-1 are negligible compared to other energy costs.

Protocol Extensions

The basic Rich Uncle protocol can be extended to establish a key with three or more single-hop-connected SAFEMITS nodes, instead of just pairs. Such a *group* Rich Uncle protocol will provide even greater gains over multiple pairwise key establishments.

Similarly, the Rich Uncle method can be extended to establish a common key with two or more multi-hop-connected SAFEMITS nodes, instead of just singly-hop-connected nodes. Although such an extension allows more nodes to gain the benefits of using the Rich Uncle method, it does so at the expense of additional communications required to forward key management information over multiple hops. Depending on the relative cost of computation and communications within participating SAFEMITS nodes, Rich Uncle benefits generally do not extend beyond two or three hops from the super node.

Further Protocol Extensions[26]

The protocols presented above appeared in the draft version of this document (dated June 1, 2000) subsequently we have developed versions of the protocol to provide greater energy efficiency / security. The enhancements consist of modifications to the "Message 3" portion of the original protocol.

Improving the Energy Efficiency of the Rich Uncle protocol

In the new step 3 (enhancement 1) of the protocol, each participating node sends the super node its contribution to the shared key using the following message the first time:

$$Message3e = MsgID \,||\, Cert_i \,||\, E(S_{pub}, Counter \,||\, ID_i \,||\, ID_j \,||\, K_i \,||\, N_i \,||\, k_{i/S})$$
$$MAC(K_i, MsgID \,||\, Cert_i \,||\, Counter \,||\, ID_i \,||\, ID_j \,||\, N_i)$$

where,

$MsgID, Counter, ID_i, K_i, N_i\ ID_j\ Cert_i\ S_{pub}$ are as described above,

$k_{i/S}$ is a symmetric key that SAFEMITS node ID_i will use to protect traffic that is sends to super node S.

Later communications between the SAFEMITS node and the super node can utilize the more efficient format

$$Message3e' = MsgID \,||\, E(k_{i/S}, Counter \,||\, ID_i \,||\, ID_j \,||\, K_i \,||\, N_i) \,||$$
$$MAC(K_i, MsgID \,||\, Cert_i \,||\, Counter \,||\, ID_i \,||\, ID_j \,||\, N_i)$$

As in the original version of this protocol the new Step 3 provides the super node with a unique proof of the node's credentials, contribution to the shared key, the identity of the other node, and a key for protecting key material sent back to node i by the super node. The node's credentials are established with the super node by providing the K_i corresponding to the $H(K_i)$ contained in the verifiable $Cert_i$. Node i's contribution to the shared key is $K_i \,||\, N_i$. The MAC provides binding integrity between all message parameters.

In all versions of the protocol the certificate $Cert_i$ does not need to be send to the super nodes each time SAFEMITS node i needs to establish a pairwise or group key with another node(s), *if* the super node will reliably store the necessary information from the certificate. Storing this information has some security implications, see below.

In this version of the protocol $E(S_{pub}, Counter \,||\, ID_i \,||\, K_i \,||\, N_i \,||\, k_{i/S})$ only needs be sent to the super node once, until the key $k_{i/S}$ needs to be changed, *if* the super node will reliably store the key $k_{i/S}$. Comparing the original version of the protocol with this version we see that the initial step 3 message from a SAFEMITS node to a super node is slightly longer and more computationally expensive in the new version of the protocol but if the SAFEMITS node need to establish 2 or more keys then is version of the protocol offer substantial computational and communication energy savings.

Table 27 presents the modified protocol's energy consumption for four SAFEMITS node configurations. These values where obtained by using Message 3e' and dropping Step 1 since the super node's certificate was distributed earlier and using the same assumptions about message component sizes as above.

---

[26] Based on the assumption that most SAFEMITS nodes will need to set up more that one shared key.

Node Composition	SAFEMITS Node Energy Consumed (mJ)	Super Node Energy Consumed (mJ)	Energy Ratio, Super/SAFEMITS
R4000 Processor w/ WINS Communications	28.45	30.46	1.07
R4000 Processor w/ MuRF Communications	7.80	7.71	0.99
Dragonball Processor w/ WINS Communications	28.45	30.46	1.07
Dragonball Processor w/ MuRF Communications	7.80	7.71	0.99

Table 27 - Relative Energy Consumption for (Modified) Rich Uncle Scenarios

The values are a substantial saving over the standard version, though the energy shifting disappears. However, Table 27, only presents the cost of the protocol when the Message 3e' message is used. The Message 3e message must be used at a minimum when a SAFEMITS node first contacts a super node. The protocol using the Message 3e message energy consumption is very similar to the earlier Rich Uncle protocol. Table 28 shows the average energy consumption per protocol run if a SAFEMITS node uses a super node to establish six key pairs (assuming the certificates are only distributed as needed).

Node Composition	SAFEMITS Node Energy Consumed (mJ)	Super Node Energy Consumed (mJ)
R4000 Processor w/ WINS Communications	39.77	55.76
R4000 Processor w/ MuRF Communications	10.97	16.61
Dragonball Processor w/ WINS Communications	53.55	193.02
Dragonball Processor w/ MuRF Communications	24.76	153.86

Table 28 – Average Energy Consumption for (Modified) Rich Uncle (6 key-pairs)

In this situation the modified Rich Uncle saves between 17% (Dragonball-MuRF-Super Node) and 41% (R4000-WINS-SAFEMITS Node) of the cost of the original version of the protocol.

Improving the Security of the Rich Uncle protocol

One security problem that occurs to varying degrees in both of the above versions of the Rich Uncle protocol is that a (compromised) super node can impersonate any SAFEMITS node that utilizes it to another super node. Running the above protocols requires a SAFEMITS node to divulge its secret key K_i to the super node.

One approach that addresses this problem would be for each SAFEMITS node to have a different for K_i and corresponding certificate each super node. This approach reduces the flexibility advantage that Rich Uncle protocols have over the IDSK protocol and we reject it. An alternative approach is to "hash" (actually a one-way function[Menezes97) each SAFEMITS node's K_i multiple times and store the result, $H^n(K_i) = H(H(H(\ldots (H(K_i))\ldots)))$. in the SAFEMITS node's certificate $Cert_i$ and change Message 3 to:

$$Message3s = MsgID \,||\, Cert_i \,||\, E(S_{pub}, Counter \,||\, ID_i \,||\, ID_j \,||\, H^{n-a}(K_i) \,||\, N_i \,||\, k_{i/S}) \,||\, MAC(K_i, MsgID \,||\, Cert_i \,||\, Counter \,||\, ID_i \,||\, ID_j \,||\, N_i)$$

where,

$a(i)$ is positive number less than n.

Later communications between the SAFEMITS node and the super node can utilize the more efficient format

$$Message3s' = MsgID \,||\, E(k_{i/S}, Counter \,||\, ID_i \,||\, ID_j \,||\, H^{n-a}(K_i) \,||\, N_i) \,||\, MAC(K_i, MsgID \,||\, Counter \,||\, ID_i \,||\, ID_j \,||\, N_i)$$

The trick here is to synchronize the value of a between the SAFEMITS node and the super node(s) and to increase it over time so that each values of $H^{n-a}(K_i)$ that have been used in the past become useless to an adversary.

One approach would be establish a "time zero" (included in each certificate) that specifics when a has the value of zero for the SAFEMITS node. A DSN would have a policy of incrementing the value of each a every hour or day. The value n would have to be sufficiently large such that each $a < n$ for the lifetime of the SAFEMITS node. This approach will require that $E(S_{pub}, ID_i \,||\, H^{n-a}(K_i) \,||\, k_{i/S})$ will need to be recalculated and sent more frequently than in Message 3' above.

This approach need not consume no more energy than the improved version of the Rich Uncle protocol, the hashes $H(K_i), H(H(K_i)), \ldots, H^n(K_i)$ in a table on each SAFEMITS node. The cost of computing one of these hashes is low, see Table 8, time-memory trade-offs can be used to balance computational cost vs. storage cost.

5.3.6 Public Key Based Kerberos Protocols

Most key establishment protocols designed for mobile /wireless environments establish a key shared by a single mobile device and some base station (or central service). These protocols would not be suitable for establishing a shared key between two mobile devices. However a number of other public-key based key establishment protocols that are arbitrated by trusted servers have been developed in other settings. In this section we examine the suitability of these Public Key Based Kerberos protocols in a Distributed SAFEMITS Network. The protocols of Tung et al [Tung00a], Sirbu[Sirbu97] and Davis[Davis95],[27] the so called public key based initial authentication protocols (PKINT) [Tang00a] could be used within a DSN. Other "public key Kerberos systems such as Tung et al [Tung00b] only employ public key cryptography for cross realm operations which roughly corresponds to operations between different DSNs. Such operations are beyond the scope of this report and these protocols will not be discussed further here.

---

[27] The related Needham and Schroeder [Needham78] protocol uses seven messages and therefore has higher latency than the protocols presented in this section. In addition a couple of flaws have been discovered in the protocol. Denning and Socco [Denning81] discovered a replay attack against the protocol that can be overcome in various ways including using timestamps. The original protocol also contains a flaw, discovers by Lowe [Lowe95], that allows an adversary to impersonate one SAFEMITS node to another, which can be prevented by including additional identity information in protocol. Therefore, we rule out using this protocol in distributed SAFEMITS networks.

It is interesting to note that the primary focus on the efficiency of the PKINT protocols is on server scalability and state minimization rather than client computational efficiency or even communication minimization (as measured by the number of bits transmitted and received). Also note that the protocols are limited to two party communication establishment rather than providing for small groups.

Davis' Public Key Kerberos

Davis' PK Kerberos protocol utilizes symmetric key cryptography to establish a shared session key between the initiating client (Node A) and the Kerberos server and to distribute the shared symmetric key between the clients, Node A and Node B. The Kerberos server only uses public key cryptography to distribute the shared session key to Node B. This design requires that secret key shared by each SAFEMITS nodes and the Kerberos server already be in place before the protocols commences. In a DSN setting such a protocol does not provide us with flexibility that we look for from a public key based protocol. We rule out protocols (such as this) that burden a DSN with the increase energy cost of public key cryptography or symmetric key cryptography without providing sufficient gains in flexibility and perhaps security as well.

Sirbu and Chuang's Public Key Kerberos

Sirbu and Chang take a very different approach to employing Public Key cryptography in Kerberos. They do away with the authentication server altogether converting initial authentication and session key establishment between Node A and Node B into a two party protocol. This protocol is not arbitrated (it is a form of pairwise asymmetric keying) and therefore cannot utilize the arbitrator's enhanced capabilities such as greater battery energy to reduce the energy load that key establishment places on the SAFEMITS nodes. The protocol does improve upon the techniques described in Section 5.4.1. We will not discuss this protocol further.

Tang et al.'s Public Key Initial Authentication Kerberos

Tang et al.'s protocol (PKINT) requires that the clients (SAFEMITS nodes) perform expensive public key operations including signing messages. Using this protocol in a DSN would place a significantly greater energy load on the SAFEMITS nodes that the Rich Uncle protocol discussed above. Therefore we rule out this protocol as well.

5.4 Self-Enforcing Autonomous Keying Protocols

5.4.1 Pairwise Asymmetric Keying

A large number of pairwise key establishment protocols have been developed since the announcement of the Diffie-Hellman Key Agreement Protocol in 1976. Many of these protocols (including the Diffie-Hellman) unaltered are not suitable use in distributed SAFEMITS networks since they do not provide sufficient entity authentication.

In this section we describe a class of pairwise asymmetric protocols (protocols that encrypt and sign a session key, [Menezes97, p509]) and discuss their energy consumption. Then we compare energy cost of this class of protocol against a Beller-Yacobi protocol that was designed to reduce the computational cost (number of CPU cycles) of one of the protocols participants (i.e. a smartcard).

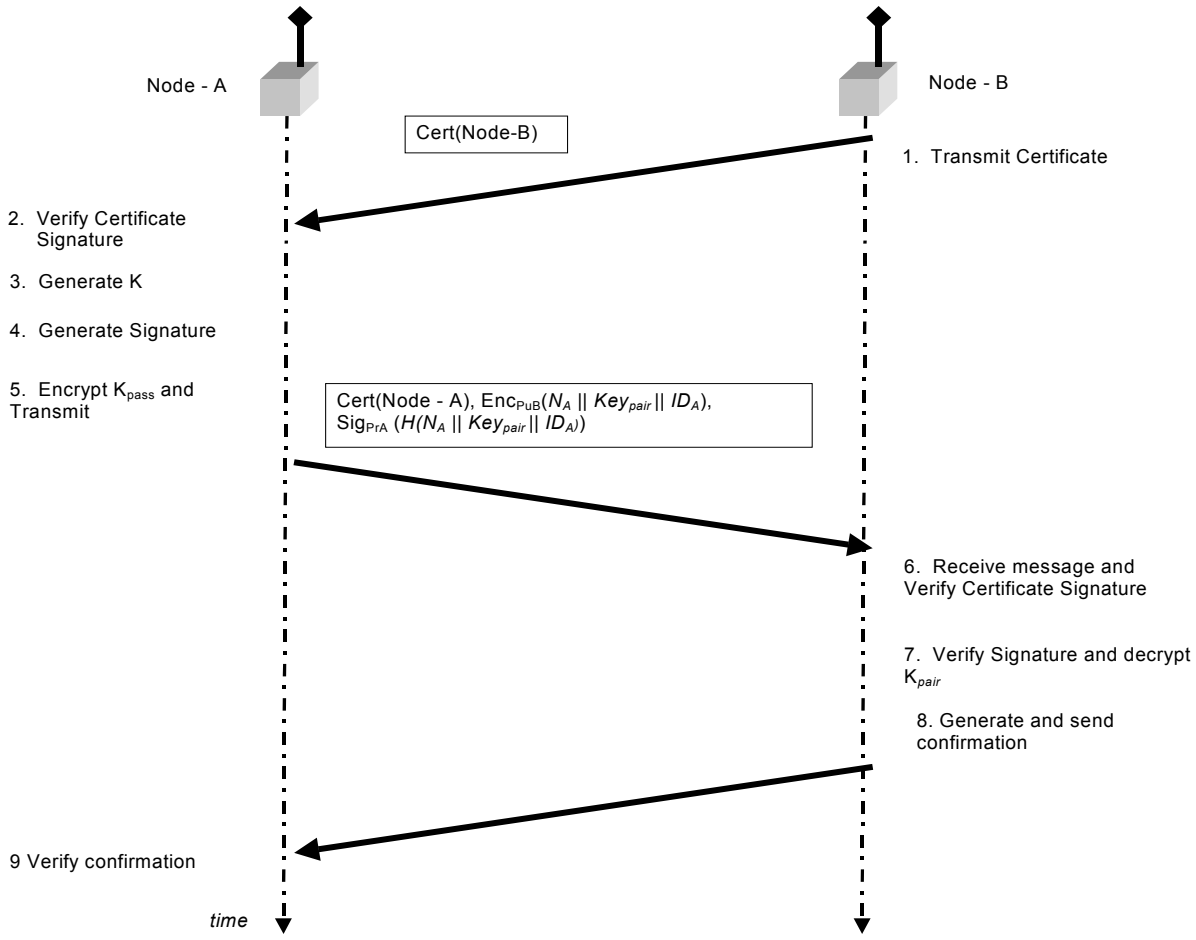

Figure 16 – Pairwise Public Key Based Protocol (PK-TPP)

The Protocol

Each SAFEMITS node such as Node A, has a public-key certificate $cert_A$ which provides a copy of Node A's public key and provides a binding between the key and Node A's identity, ID_A. The public-key signature contained within the certificate provides this binding. The Mission Authority using its private key, $K_{MA\text{-}private}$, generates this signature. Each SAFEMITS node has embedded within it during the pre-deployment phase the corresponding public key, $K_{MA\text{-}public}$, which it uses to verify certificates signed by the Mission Authority.

The protocol begins with the initiator SAFEMITS node, Node B, generating a random nonce, N_B, and sending it along with its certificate to Node A.

Round 1

Node B → Node A: $ID_A \| ID_B \| N_B \| Cert_B$

Node A receives the message and verifies node B's certificate (i.e. verifies the signature generated by the Mission Authority). Node A generates the shared key Key_{pair} and a random nonce N_A. Node A encrypts N_A and Key_{pair} so that the key and the nonce will be communicated to Node B privately. Node A generates $Sig(K_{A\text{-}private}, Hash(N_A \| Key_{pair} \| ID_A))$ which will provide Node B with an authenticated copy of the key and the nonce.

Node A sends its certificate, signature and the encrypted value to Node B

Round 2

Node A → Node B: $ID_B \,||\, ID_A \,||\, E(K_{B\text{-}public},\ N_A \,||\, Key_{pair} \,||\, ID_A) \,||\, Sig(K_{A\text{-}private},\ Hash(N_A \,||\, Key_{pair} \,||\, ID_A)) \,||\, Cert_A$

Node B verifies Node A's certificate, decrypts the shared key using its private key $K_{B\text{-}private}$, and verifies the signature using Node A's public key, $K_{A\text{-}public}$. Now only Node A and Node B know the shared key Key_{pair} and the nonce N_A.

Optionally Node B can prove to Node A that it did participate in the protocol (the Round 1 message could be a replay) and that it received and successfully obtained the key by sending the following message back to Node A.

Round 3

Node B → Node A: $ID_A \,||\, ID_B \,||\, E(Key_{pair},\ N_A)$

Node A decrypts the message and checks that the nonce N_A is correct. The protocol can be realized using different combinations of public-key algorithms such as RSA, ElGamal encryption or signature, Elliptic Curve encryption, XTR encryption or signature, or DSA. The full version of the protocol provides mutual entity authentication and key delivery confirmation. The short version of the protocol only provides one-way entity authentication and must rely on the use of Key_{pair} by Node B in order for Node A to know that Node B has successfully received the key.

The energy cost of this protocol and how the energy cost of the protocol is distributed between the participating SAFEMITS nodes varies with the choice of crypto-algorithm. Table 29 presents the cost of this protocol with the optional third message for protocols using the RSA and XTR cryptosystems.

As part of our research we have been examining the Beller-Yacobi protocols. These protocols are public-key based pairwise key establishment protocols. The most efficient of these protocols is the two-pass protocol. The Beller-Yacobi two-pass protocol (BY-2) [Beller93] (typically) employs RSA encryption and either ElGamal or DSA signature methods plus a symmetric key method. It was designed to reduce the computational (# CPU cycles) cost for one of its participants (a smartcard or a wireless terminal) interacting with a higher power peer. However, when we examined the energy consumption of the protocol a different picture appeared. The energy consumed by the participants differs by less than 10% from the energy consumed by either node and the total energy consumed was nearly twice the energy consumed by the protocol presented in this section.

Processor		Total Energy Consumption (mJ)	
		RSA	XTR
MIPS R4000	Comm.	236	116
	Comp.	36	17
	Total	272	133
SA-1110 "StrongARM"	Comm.	236	116
	Comp.	32	16
	Total	269	132
Z-180	Comm.	236	116
	Comp.	7,923	3,884
	Total	8160	4,000
MC68328 "Dragon Ball"	Comm.	236	116
	Comp.	1,807	886
	Total	2,043	1,002
MC68328 "ColdFire"	Comm.	236	116
	Comp.	1,667	817
	Total	1,904	933
MMC2001 "M-Core"	Comm.	236	116
	Comp.	296	145
	Total	532	261
ARC-3	Comm.	236	116
	Comp.	2	1
	Total	239	117

Table 29 - Energy Consumption of Pairwise Public Key Protocol

5.4.2 Group Keying Protocols

In this section we will look at the effectiveness of techniques for establishing a shared group key among a group of DSN SAFEMITS nodes without using a special trusted third party (such as a super node). These methods, like the pairwise methods in the previous section, are all based on public key cryptography techniques, primarily the use of modular exponentiation over Z_N (RSA) or a finite field such as Z_P (Diffie-Hellman).

5.4.2.1 (Elected) Simple Key Distribution Center

This protocol is designed to support small groups of nodes using only unicast messages. It is designed for efficiency and does not provide perfect forward secrecy (break back protection). If an adversary compromises a group leader node he can read past traffic and obtain past group keys for those groups for which the compromised node acted as the group leader. However, for DSN operations, such as routing, this limitation is tolerable. In this protocol we utilize a pairwise key establishment protocol to distribute the group key securely and bootstrap the distribution of the (initial) group key.

The Protocol

Each SAFEMITS node, such as Node A, has a public-key certificate $cert_A$ which provides a copy of Node A's public key and provides a binding between the key and Node A's identity, ID_A. The public-key signature contained within the certificate provides this binding. The Mission Authority using its private key, $K_{MA\text{-}private}$, generates this signature. Each SAFEMITS node has embedded within it

during the pre-deployment phase the correspond public key, $K_{MA\text{-}public}$, which it uses to verify certificates signed by the Mission Authority. We also assume that each nodes know a DSN wide key $Key_{(Ni,Nj)}$.

Before the protocol begins, we assume that the membership of the group has been established and that each node in the group knows that a particular node, Node J, will act as the group leader.

The protocol begins with Node J sending its certificate to the other group members.

Round 1

Node J → Node * (all other group members)**:**

$$ID^* \,||\, ID_J \,||\, Cert_J$$

Each recipient node (Node I) verifies Node J's certificate (i.e. verifies the signature generated by the Mission Authority).

Each Node I generates a shared key $Key_{(Ni,Nj)}$ and a random nonce N_I. Node I encrypts N_I and $Key_{(Ni,Nj)}$ so that the key and the nonce will be communicated to Node J privately. Node I generates $Sig(K_{I\text{-}private}, Hash(N_I \,||\, Key_{(Ni,Nj)} \,||\, ID_I))$ or $MAC(Key_{(Ni,Nj)}, N_I \,||\, Key_M \,||\, ID_I \,||\, H(membership))$. The encrypted value and the signature or MAC will provide Node J with an authenticated copy of the shared key and the nonce. $H(membership)$ is a hash of a list of the members of the group with the group leader's ID first followed by the other member's IDs in ascending numerical order (and perhaps including a group specific counter).

If Node J has been the group leader before for every node in the group then Round 2 can be skipped.

Node I then sends either of the following Round 2 messages (depending on the security policy of the DSN) to Node J.

Round 2

Node I → Node J: $ID_J \,||\, ID_I \,||\, E(K_{J\text{-}public}, N_I \,||\, [\, Key_{(Ni,Nj)} \,||]\, ID_I \,||H(membership)) \,||\, MAC(Key_{(Ni,Nj)}, N_I \,||\, Key_{(Ni,Nj)} \,||\, ID_I \,||\, H(membership))$

or

Round 2'

Node I → Node J: $ID_J \,||\, ID_I \,||\, E(K_{J\text{-}public}, N_I \,||\, [\, Key_{(Ni,Nj)} \,||]\, ID_I||H(membership)) \,||\, [\, Cert_I \,||]\, Sig(K_{I\text{-}private}, H(N_I \,||\, Key_{(Ni,Nj)} \,||\, ID_I \,||\, H(membership)))$

Round 2 continues with Node J decrypting $E(K_{J\text{-}public}, N_I \,||\, [\, Key_{(Ni,Nj)} \,||]\, ID_I \,||H(membership))$ and verifying ID_I and $H(membership)$. Node J then uses $Key_{(Ni,Nj)}$ to verify $MAC(Key_{(Ni,Nj)}, N_I \,||\, Key_{(Ni,Nj)} \,||\, ID_I \,||\, H(membership))$. Node J then determines that the sender of the message is a valid SAFEMITS node (assuming that Key_M has not been compromised) but cannot be sure that the sender is Node I.

Alternatively in Round 2' Node J verifies each Node I's certificate, decrypts $E(K_{J\text{-}public}, N_I \,||\, [\, Key_{(Ni,Nj)} \,||]\, ID_I \,||H(membership))$ using its private key $K_{J\text{-}private}$, and checks its contents. Node J also verifies the signature using each Node I's public key, $K_{I\text{-}public}$.. The verification process includes checking $H(membership)$. Node J can be sure that the message sender is Node I (unless Node I's private key has been compromised). Now only each Node I and Node J know the shared key $Key_{(Ni,Nj)}$ and the nonce N_I.

Multiple groups may be formed involving the same pair of nodes, (Node I and Node J) with the same node as group leader. In such a situation the optional portions of this message can be

eliminated from later runs of the protocol. Also note that once Round 2' has been performed by a sender and a receiver Round 2 can be used later, with the same key $Key_{(Ni,Nj)}$ or a symmetric key encryption function can be used to replace $E(K_{J\text{-}public}, \ldots)$. Section 5.3.5.1(Energy Consumption) demonstrates how this can be done.

Node J now can generate the group key, Key_G. The key generation process is not part of the protocol, one approach would be for Node J to generate a random nonce N_J and use all of the nonces as inputs to some key generation function, (e.g. $Key_G = F(N_J, N_A \,||\, N_B \,||\, \ldots \,||\, N_E \,||\, N_F)$). Node J then distributes the group key to the other group members.

Round 3

Node J → Node I (for each group member other than J):

$$ID_I \,||\, ID_J \,||\, E(Key_{(Ni,Nj)}, N_I \,||\, Key_G. \,||\, H(membership))$$

Each Node I decrypts "its" message and checks that the nonce N_I and the hash of the group membership are correct.

At this point each group member should know the group key but the other group members have not confirmed this fact. The simplest approach is for each group member to tell the group leader that it has received the group key and for the group leader to inform the entire group when all the members have confirmed having the group key. Each node confirms to the group leader that it has received the group key by encrypting the nonce N_I and a hash of the group key the membership and nonce to the group leader.

Round 4

(each) Node I → Node J: $ID_J \,||\, ID_I \,||\, E(Key_{(Ni,Nj)}, N_I \,||\, H(N_I \,||\, Key_G \,||\, H(membership)))$

Node J decrypts each message and determines that each Node I received the key for the proper group. Node J can inform the group that the group key is in place by generating and sending either:

Round 5

Node J → Node I*:

$$ID^* \,||\, ID_J \,||\, E(Key_{Group}, N'_J \,||\, H(N'_J \,||\, Key_G \,||\, H(membership)))$$

or

Round 5'

Node J → Node I*:

$$ID^* \,||\, ID_J \,||\, N'_J \,||\, Sig(K_{I\text{-}private}, Hash(N'_I \,||\, Key_G \,||\, H(membership)))$$

to the other group members who can decrypt and verify the message contents. The Round 5 message is cheaper to generate, transmit, receive, and verify. Using the Round 5' message prevents any of the group members from spoofing the other group members into believing than Node J says that the group key is in place. Round 5' should not be used with Round 2.

Like the Pairwise Public Key protocol this protocol can be realized using a number of different public-key algorithms such as RSA, ElGamal encryption or signature, Elliptic Curve encryption, XTR encryption or signature, or DSA. The energy cost of this protocol and how the energy cost of the protocol is distributed between the participating SAFEMITS nodes varies with the choice of processor, communication subsystem and crypto-algorithm and the protocol "variant" chosen.

The following tables present energy consumption of the two variants of the protocol using RSA both public-key encryption and for signatures. Under these assumptions:

- The group size is 6.
- The node ID sizes are 32 bits.
- The symmetric keys are 128 bits.
- All nonces are 64 bits.
- RSA modulus size is 1024 bits.
- The node certificate size is 2500bits.
- MAC and hash sizes are both 128 bits.
- The Round 1 message is 2664 bits.
- The Round 2 / 2' messages are 576 / 3972 bits.
- The Round 3 message is 448 bits.
- The Round 4 message is 320 bits.
- The Round 5 / 5' messages are 320 / 1088 bits.

In the "lightweight" version of the protocol the group leader consumes 4 times as much energy as do each of the ordinary nodes (3 times as much in the "heavy weight" version) performing communications. The lightweight protocol leader also consumes nearly 50 times as much energy performing computations (5 times as much in the "heavy weight" version) for this group size. The total energy consumed by all of the participants in the heavy weight version of the protocol is between 2 1/5 and 2 1/3 times larger than that consumed by the participants in the lightweight version of the protocol.

Processor		Energy Consumption (mJ)		
		Standard Node	Leader Node	Total
MIPS R4000	Comm.	65	170	498
	Comp.	2	84	92
	Total	67	254	590
SA-1110 "StrongARM"	Comm.	65	170	498
	Comp.	2	75	83
	Total	67	245	580
Z-180	Comm.	65	170	498
	Comp.	372	18,431	20,291
	Total	438	18,601	20,789
MC68328 "Dragon Ball"	Comm.	65	170	498
	Comp.	84	4201	4624
	Total	150	4372	4268
MCF5204 "ColdFire"	Comm.	65	170	498
	Comp.	78	3879	4269
	Total	143	4048	4765
MMC2001 "M-Core"	Comm.	65	170	498
	Comp.	14	688	757
	Total	79	858	1254
ARC-3	Comm.	65	170	498
	Comp.	0	6	6
	Total	66	176	503

Table 30 - Energy Consumption of (Elected) Simple Key Distribution Center (Using Rounds 2 and Round 5)

Processor		Energy Consumption (mJ)		
		Standard Node	Leader Node	Total
MIPS R4000	Comm.	148	424	1162
	Comp.	19	104	200
	Total	167	529	1362
SA-1110 "StrongARM"	Comm.	148	424	1162
	Comp.	17	94	180
	Total	165	518	1341
Z-180	Comm.	148	424	1162
	Comp.	4240	23,035	44,235
	Total	4387	23,460	45,397
MC68328 "Dragon Ball"	Comm.	148	424	1162
	Comp.	968	5253	10,087
	Total	1114	5677	11,248
MCF5204 "ColdFire"	Comm.	148	424	1162
	Comp.	892	4847	9308
	Total	1040	5271	10,470
MMC2001 "M-Core"	Comm.	148	424	1162
	Comp.	158	860	1651
	Total	306	1284	2812
ARC-3	Comm.	148	424	1162
	Comp.	1	7	14
	Total	150	431	1175

Table 31 - Energy Consumption of (Elected) Simple Key Distribution Center (with Rounds 2' and 5')

5.4.2.2 Cliques Group Diffie-Hellman Protocols

The Cliques Group Diffie-Hellman keying protocols are a set of key agreement protocols (each groups provably contributes to the value of the group key) that have been developed since the early 1990's. We focus on those methods that include authentication (since we assume the presence of active adversaries) and we have inserted certificate exchange into the protocols to better *model* the ad-hoc nature of DSNs.

The setting

Assume that a group of neighboring DSN forms a group to efficiently protect message amongst themselves (e.g. for collaborative signal processing) and for distributing / forwarding traffic from sources outside the group and therefore the group key will be put into use shortly after creation. Furthermore assume that some other (previously executed) protocol allows the members of the group to know the "identity" of the other members of the group (*this is not a trivial point*). Each member of the GDH protocol we will describe below needs to know who to send the its output to (i.e. who goes first, second … last) and in the case of two of the nodes the complete membership of the group. Assume that each participant has a public key of the form $\alpha^{x_i} \pmod{p}$ signed using the RSA key of the "owner" of the DSN. Each node provides its certificate, as part of the protocol, to the group leader who is trusted by the others with the function of controlling access to the group.

We now describe and analyze the cost of Ateniese, Steiner and Tsudik's A-GDH.3 protocol [Ateniese99], an authenticated Group Diffie-Hellman Keying protocol that was designed to minimize

computational cost and total bandwidth, modified with the addition of public key certificates. A-GDH.3 is an extension (providing authentication) of the GDH.3 / IKA.2 protocol by the same authors.

The protocol

The protocol consists of n rounds; n is the size of the group. Each node *i* has a certificate $cert_i$, which contains the identity of the node and its public key $\alpha^{x_i} \pmod p$ signed using the network's owner RSA key. Let p be a prime and q a prime divisor of $p-1$. G is a cyclic subgroup of $Z_P - \{0\}$ of order and α is a generator of G. The values p, q and α are known (and known to be authentic) to all of the nodes of the DSN. This can be done efficiently by installing these values in each DSN node prior to deployment. These values do not need to be kept confidential.

Round i (0 < i < n-1)

Node *i* → Node *i+1*: $ID_I \,||\, ID_{I+1} \,||\, \alpha^{(r_1 \ldots r_i)} \pmod p$

Each node takes a value (node *1* starts with α) in $Z_P\text{-}\{0\}$ provided by its predecessor and raises it by a unique random value r_i in G chosen by that node. That node sends the result to the next node.

Round n-1

Node *n-1* → Node *i*: $ID_{N-1} \,||\, ID_i \,||\, \alpha^{(r_1 \ldots r_{n-1})} \pmod p$

In this round Node *n-1* takes the values provided by Node *n-2* and raises it by a unique random value in G chosen by Node *n-1* and sends the result to Node *1*, Node *2*, … Node *n-2*.

Round n

Node *i (0 < i < n)* → Node *n*: $ID_I \,||\, ID_N \,||\, \alpha^{(r_1 \ldots r_{n-1})/\, r_i} \pmod p \,||\, Cert_i$

In this round Node *1*, Node *2*, … Node *n-1* each take the value received in Round *n-1* and raises it by the inverse of their unique random value r_i. That result and the certificate of the node are sent to Node *n*.

Round n + 1

Node *n* → Node *i*: $ID_{N-1} \,||\, ID_i \,||\, \{\alpha^{((r_1 \ldots r_n)/\, r_i) K_{in}} \pmod p \mid 1 \le i \le n-1\} \,||\, Cert_n$

In this round Node *n* verifies each certificate and raises each value received by a unique random value r_n in *G* chosen by Node *n* and by $K_{in} = F(\alpha^{x_i x_n} \pmod p)$. F is a lightweight bijection function mapping elements of G into Z_q. Note that Node *n* alone of all the DSN nodes knows x_n and uses that value and the public keys of the other nodes to determine each K_{in}.

After they receive their message from Node *n*. Node *1*, Node *2*, … Node *n-1* each independently verifies the certificate of Node *n*. Each node then takes the value provided to it by Node n and raises the value by Node *i*'s unique random value r_i in G chosen earlier and the inverse of K_{in}. Node *i* alone of all the DSN nodes knows x_i and uses that value and Node *n*'s public key $(\alpha^{x_i} \pmod p)$ to determine K_{in} and its inverse. The result $\alpha^{(r_1 \ldots r_n)}$ is the shared group key.

The following table provides estimated energy cost of this protocol for a group of 6 nodes when only unicast messages are available. These calculations again assume that a WINS transceiver is used and the inter-node distance is 100 meters.

Processor		Energy Consumption (mJ)			
		Nodes 1-4	Node 5	Node 6	Total
MIPS R4000	Comm.	179	232	643	1575
	Comp.	64	33	99	390
	Total	243	265	743	1965
SA-1110 "StrongARM"	Comm.	179	232	643	1575
	Comp.	59	30	91	357
	Total	238	262	734	1933
Z-180	Comm.	179	232	643	1575
	Comp.	14,725	7455	22,735	89,092
	Total	14,904	7687	23,378	90,668
MC68328 "Dragon Ball"	Comm.	179	232	643	1575
	Comp.	3,358	1700	5185	20,319
	Total	3537	1933	5828	21,895
MC68328 "ColdFire"	Comm.	179	232	643	1575
	Comp.	3099	1569	4784	18,748
	Total	3278	1801	5427	20,324
MMC2001 "M-Core"	Comm.	179	232	643	1575
	Comp.	550	278	848	3325
	Total	728	510	1492	4901
ARC-3	Comm.	179	232	643	1575
	Comp.	5	2	7	27
	Total	183	234	650	1603

Table 32 - Energy cost of A-GDH.3 including certificates with unicast messages only

The energy burden on the nodes is unevenly distributed in this protocol. Both the computational energy and communication energy are unevenly distributed so switching from the WINS transceiver to the MuRF transceiver would not change this imbalance.

Compared to the unauthenticated version of this protocol (IKA.2) this protocols is significantly more expensive for most processors when using the WINS transceiver. For the MIPS R4000 there is a 84% increase in total energy cost when using the authenticated version of the protocol (with certificate transport), whereas for the ARC-3 processor there is more than a 125% increase in the energy consumed.

This protocol provides key agreement, no group member can control the value of the group key, each group member can influence the value of the key. In situations where key agreement is not needed, the group leader is trusted by the other nodes to generate and distribute the group key, and the group size is small, it is more energy efficient for each group member to establish a pairwise key with the leader, as in Section 5.4.2.1, and have the group leader generate and distribute the key. A significant advantage of this protocol over the protocol from the previous section is that this protocol provides perfect-forward secrecy, see Section 3.4, and is better suited for establishing keys that protect long-term secrets.

The following table provides estimated energy cost of this protocol for a group of 6 nodes when both multicast and unicast messages are available. These calculations again assume that a WINS transceiver is used and the inter-node distance is 100 meters.

Processor		Energy Consumption (mJ)			
		Nodes 1-4	Node 5	Node 6	Total
MIPS R4000	Comm.	240	224	418	1602
	Comp.	64	33	99	390
	Total	304	257	517	1991
SA-1110 "StrongARM"	Comm.	240	224	418	1602
	Comp.	59	30	91	357
	Total	299	254	509	1959
Z-180	Comm.	240	224	418	1602
	Comp.	14,725	7455	22,735	89,092
	Total	14,965	7680	23,153	90,694
MC68328 "Dragon Ball"	Comm.	240	224	418	1602
	Comp.	3358	1700	5185	20,319
	Total	3598	1925	5603	21,921
MC68328 "ColdFire"	Comm.	240	224	418	1602
	Comp.	3099	1569	4784	18,748
	Total	3339	1793	5202	20,350
MMC2001 "M-Core"	Comm.	240	224	418	1602
	Comp.	550	278	848	3325
	Total	789	503	1266	4927
ARC-3	Comm.	240	224	418	1602
	Comp.	5	2	7	27
	Total	244	227	425	1629

Table 33 - Energy cost of A-GDH.3 including certificates with both unicast and multicast messages

The use of multicast messages is slightly more expensive than only using unicast, with an increase of approximately 25 mJ for the WINS transceiver. However, the latency of this mixed message type approach is considerably lower, for example if all of the nodes in the group are one hop or less away from Node 5 and Node 6 then the latency of the unicast method is approximately 60% greater than the protocol that uses both unicast and multicast messages.

5.4.2.3 Burmester-Desmedt Conference Keying

The Burmester-Desmedt conference key protocols first appeared in [Burmester94]. In that paper a number of techniques were presented for forming a group key. The most efficient of them was designed to work best in a broadcast environment and offers better performance (energy cost) in a DSN multicast environment as well. This protocol like the GDH protocol discussed above assumes that the global parameters (p, q and α) for the underlying Diffie-Hellman method have been properly distributed to the DSN nodes by the network's owner. We also assume that each node *i* has a certificate $cert_i$, which contains the identity of the node and its public key, signed using the network owner's private RSA key. As in the A-GDH.3 analysis, the Mission Authority's public RSA

key is known to all the DSN nodes and each node initially only knows its own certificate. Unlike the GDH analysis we assume that each group member knows all of the nodes that should make up the group.[28]

An Authenticated Protocol

The protocol consists of only two rounds if multicast communications are used, where *n* is the size of the group. Each node takes α and raises it by a unique random value r_i in G chosen by that node obtaining $z_i = \alpha^{r_i} \pmod p)_i$. Each node signs the hash of the result using its public key and sends the result, the signature, and its certificate to the other nodes.

Round 1

Node *i* → Node *: $ID_i \| ID_* \| z_i \| Sig_{PubK_i}(ID_i, z_i) \| cert_i$

Each node *i* verifies the certificate and the signature the messages from its neighbors, *i-1 (mod n)* and *i+1 (mod n)*, and constructs $X_i = (\alpha^{(r_{i+1})} / \alpha^{(r_{i-1})})^{r_i} \pmod p)$ signs it and sends X_i and the signature to the other nodes. The transmission of the certificate is optional in Round 2, if the Round 1 messages are sent by Node *i* to its two neighbors then the certificate is required in Round 2, if Node *i* sends its Round 1 message to the entire group then the certificate is not needed in Round 2.

Round 2

Node *i* → Node *: $ID_i \| ID_* \| X_i \| Sig_{PubK_i}(Id_i, X_i) [\| cert_i]$

Each node, upon receiving the necessary X_i 's (one X from each of the other group members) verifiers their signatures and constructs K_i as follows:

$$K_i = (z_{i-1})^{nr_i} X_i^{n-1} X_{i+1}^{n-2} X_{i+2}^{n-3} \cdots X_{i-2} \pmod p)$$

Which for each node should be equal to:

$$\alpha^{r_1 r_2 + r_1 r_2 + r_2 r_3 + \ldots + r_n r_1} \pmod p)$$

The following tables provide estimated energy cost and latency of this protocol for a group of 6 nodes when only unicast messages are available. These calculations again assume that a WIN transceiver is used and the inter-node distance is 100 meters.

[28] This notion of needing to know the entire group membership may be a non-issue at least from an authentication point of view. If a node has a valid certificate and the signature "corresponds" with the certificate then the node is automatically accepted. If a node shows up in too may groups then action will be taken later.

Processor		Energy Consumption (mJ)	
		Per Node	Total
MIPS R4000	Comm.	955	5730
	Comp.	88	521
	Total	1042	6251
SA-1110 "StrongARM"	Comm.	79	5730
	Comp.	207	473
	Total	1034	6203
Z-180	Comm.	955	5730
	Comp.	19,568	117,407
	Total	20,523	123,136
MC68328 "Dragon Ball"	Comm.	955	5730
	Comp.	4463	26,777
	Total	5418	32,507
MC68328 "ColdFire"	Comm.	955	5730
	Comp.	4117	24,707
	Total	5073	30,436
MMC2001 "M-Core"	Comm.	955	5730
	Comp.	730	4382
	Total	1685	10,111
ARC-3	Comm.	955	5730
	Comp.	6	36
	Total	961	5766

Table 34 - Energy Consumption of Authenticated Burmester-Desmedt including certificate transport with unicast messages only

The Burmester-Desmedt protocols are perfectly fair to each group member in terms of energy consumption and for the DragonBall, ColdFire, and Z-180 processors the energy consumption of nodes performing the Burmester-Desmedt protocol (augmented with signatures) is less than the energy consumption of the leader node (Node N) in the A-GDH.3 protocol. However, the authenticated Burmester-Desmedt–UNICAST is more expensive than authenticated GDH-UNICAST this size group. (In the case of the ARC-3 authenticated Burmester-Desmedt is 2 1/2 times more costly). For inefficient processors such the Z-180 is authenticated Burmester-Desmedt is about 35% more expensive.

The following tables provide estimated energy cost and latency of this protocol for a group of 6 nodes when only multicast and unicast messages are available. These calculations again assume that a WIN transceiver is used and the inter-node distance is 100 meters.

Processor		Energy Consumption (mJ)	
		Per Node	Total
MIPS R4000	Comm.	742	4455
	Comp.	87	521
	Total	829	4976
SA-1110 "StrongARM"	Comm.	742	4455
	Comp.	79	473
	Total	821	4928
Z-180	Comm.	742	4455
	Comp.	19,568	117,407
	Total	20,301	121,862
MC68328 "Dragon Ball"	Comm.	742	4455
	Comp.	4463	26,777
	Total	5205	31,232
MC68328 "ColdFire"	Comm.	742	4455
	Comp.	4118	24,707
	Total	4860	29,162
MMC2001 "M-Core"	Comm.	742	4455
	Comp.	730	4382
	Total	1473	8837
ARC-3	Comm.	742	4455
	Comp.	6	36
	Total	749	4491

Table 35 - Energy Consumption of Authenticated Burmester-Desmedt including certificate transport with unicast and multicast messages

The energy efficiency of the authenticated Burmester-Desmedt protocol compared to authenticated A-GDH.3 (both with certificate transport) with both protocols using both unicast and multicast messages and the WINS transceiver and is similar to the situation when only unicast messages are used. The A-GDH.3 leader consumes more energy than the Burmester-Desmedt nodes but the total energy consumption of A-GDH.3 protocol is much lower for groups of this size.

The results for the unauthenticated versions of these protocols are much closer; Burmester-Desmedt using both unicast and multicast is outperformed by GDH.3 using both unicast and multicast by about 3% on the MIPS-R400 with the WINS transceiver and for the very efficient ARC-3 by 20%.

The Cliques protocol and Burmester-Desmedt protocols both provide perfect-forward secrecy. The latency of the mixed message type Burmester-Desmedt is the same as the unicast only protocol. However, neither protocol confirms that the group key is known to all other the group members.

5.4.2.4 Just-Vaudenary Multi-Party Key Agreement

Just and Vaudenary [Just96] developed a multi-party key agreement protocol by developing a generalization of the Burmester-Desmedt conference key protocols. They developed a generic construction for establishing group wide key agreement, which uses any pair-wise authenticated Diffie-Hellman key agreement protocol as a basis.

The Just-Vaudenary protocol assumes that the global parameters (p, q and α) for the underlying Diffie-Hellman method have been properly distributed to the DSN nodes by the network's owner. Each node i has a certificate cert_i, which contains the identity of the node and its public *key* $p_i = \alpha^{x_i}$

(mod p) signed using the network's owner RSA key. The Mission Authority's public RSA key is assumed to be known to all the DSN nodes and each node initially only knows its own certificate.

Protocol

The protocol consists of a pair-wise key agreement phase and the group phase. The authors present two candidate protocols for the pair-wise phase. Here we use the protocol that does not require that the nodes know their neighbor's (in the group) certificates. The size of the group is *n*.

Pairwise Phase

Round 1

Node i → Node i+1: $ID_i \,||\, ID_{i+1} \,||\, (y_i = \alpha^{x_i} \pmod p) \,||\, cert_i$

Each node takes α and raises it by a unique random value x_i in G chosen by that node. Each node sends this result and its certificate to the next highest number group member (modulo group size). The receiver takes α and raises it by a unique random value x'_{i+1} in G chosen by that node and calculates $K'_{i+1} = y_i^{(x'_{i+1}+S_{i+1})}\, p_i^{x'_{i+1}}$ and the keyed hash (MAC) $z'_{i+1} = h_{K'_{i+1}}(y'_{i+1} || \, y_i \,||\, ID_i \,||\, ID_{i+1})$. Node i+1 then sends the following message to Node i.

Round 2

Node i+1 → Node i: $ID_{i+1} \,||\, ID_i \,||\, (y'_{i+1} = \alpha^{(x'_{i+1})} \pmod p) \,||\, cert_{i+1} || \, z'_{i+1}$

Node i computes $K_i = y'^{(x_i S_i^{-1})}_{i+1}$ and checks that $z'_{i+1} = h_{K_i}(y'_{i+1} || \, y_i \,||\, ID_i \,||\, ID_{i+1})$. Node i then generates the following messages and sends it to Node i +1.

Round 3 (optional)

Node i → Node i+1: $ID_i \,||\, ID_{i+1} \,||\, (z_i = h_{K_i}(y_i \,||\, y'_{i+1} || \, ID_{i+1} || \, ID_i))$

After the pairwise phase is completed the group-wide phase begins.

Group-wide Phase

Node i computes W_i and sends it to the other nodes.

Round 1

Node i → Node *: $ID_i \,||\, ID_* \,||\, (W_i = K_i \,/\, K_{i-1})$

Each node, Node i upon receiving the W_i's calculates:

$$K = K_{i-1}^{\,t}\; W_i^{\,t-1}\; W_{i+1}^{\,t-2}\; W_{i+2}^{\,t-3} \cdots W_{i-2} \pmod p$$

Which for each node should be equal to:

$$K = K_1\, K_2\, K_3 \cdots K_t \pmod p$$

The following table provides estimated energy cost of this protocol (with the optional Round 3 message in the Pairwise Phase) for a group of 6 nodes when only unicast messages are available. These calculations again assume that a WIN transceiver is used and the inter-node distance is 100 meters.

Processor		Energy Consumption (mJ)	
		Per Node	Total
MIPS R4000	Comm.	1154	6,926
	Comp.	227	1,360
	Total	1,381	8,256
SA-1110 "StrongARM"	Comm.	1154	6,926
	Comp.	207	1,243
	Total	1,361	8,169
Z-180	Comm.	1154	6,926
	Comp.	51,545	309,269
	Total	52,699	316,195
MC68328 "Dragon Ball"	Comm.	1154	6,926
	Comp.	11,756	70,535
	Total	12,910	77,461
MC68328 "ColdFire"	Comm.	1154	6,926
	Comp.	10,847	65,082
	Total	12,001	72,008
MMC2001 "M-Core"	Comm.	1154	6,926
	Comp.	1,924	11,542
	Total	3,077	18,468
ARC-3	Comm.	1154	6,926
	Comp.	16	95
	Total	1,170	7,021

Table 36 - Energy Consumption of the Just-Vaudenary Protocol including certificates with unicast messages only

The Just-Vaudenary protocol like the authenticated Burmester-Desmedt protocol is perfectly fair to each group member and in terms of energy consumption Just-Vaudenary is more efficient but is inferior to the A-GDH.3 protocol. Even when the signature and certificate are removed from Round 2 of the Burmester-Desmedt protocols the Just-Vaudenary has lower energy consumption.

The following table provides estimated energy cost of this protocol (with the optional Round 3 message in the Pairwise Phase) for a group of 6 nodes when both multicast and unicast messages are available. These calculations again assume that a WINS transceiver is used and the inter-node distance is 100 meters

Processor		Energy Consumption (mJ)	
		Per Node	Total
MIPS R4000	Comm.	828	4,966
	Comp.	227	1,360
	Total	1,054	6,326
SA-1110 "StrongARM"	Comm.	828	4,966
	Comp.	207	1,243
	Total	1,035	6,209
Z-180	Comm.	828	4,966
	Comp.	51,545	309,269
	Total	52,373	314,235
MC68328 "Dragon Ball"	Comm.	828	4,966
	Comp.	11,756	70,535
	Total	12,584	75,501
MC68328 "ColdFire"	Comm.	828	4,966
	Comp.	10,847	65,082
	Total	11,675	70,048
MMC2001 "M-Core"	Comm.	828	4,966
	Comp.	1,924	11,542
	Total	2,751	16,509
ARC-3	Comm.	828	4,966
	Comp.	16	95
	Total	844	5,061

Table 37 - Energy Consumption of the Just-Vaudenary Protocol including certificates with both multicast and unicast messages

The energy efficiency of the authenticated Just-Vaudenary protocol with both multicast and unicast messages versus significantly better than the authenticated Burmester-Desmedt protocol but is inferior to the A-GDH.3 protocol. The Just-Vaudenary protocol does not confirm that the group key is known to all other the group members.

5.4.3 Attribute-Based Keying

Attribute-based keying uses one-way functions in a manner similar to Clueless Agents [Schneier2], where only nodes that have attributes matching a sender's query would be capable of decrypting a given message. Attributes include distinguishing characteristics such as SAFEMITS capabilities or location. Energy-efficient hash algorithms, not energy-consumptive public key algorithms, perform the one-way functions of attribute-based keying.

For instance, a SAFEMITS node could construct a message that could only be correctly decrypted if the recipient was located within a square 100 meters on a side. The recipient's location would be used as a variable in the computation. The following example message could be constructed and sent:

E (K, Message) || LocationAttributeID || Resolution || Checksum || Nonce

where

E (K, Message) is the message encrypted using the location-based key,

LocationAttributeID and *Resolution* indicate the receiving node's location should be used to compute the key as follows:

K = H(Nonce || Location rounded to Resolution)

A receiving node can check that if it is in a location intended by the sender by computing:

Checksum = H(Location rounded to Resolution)

Such a method would be effective at establishing a key with all nodes within a given area, such as within a single hop away.

Unfortunately, such a method does not provide much cryptographic protection. An adversary likely knows the location of the SAFEMITS if it can receive the transmission. Even if it doesn't know its exact location, an adversary will know the location with sufficient accuracy to "brute force" guess proximate location values until the correct position is found. Similarly, other distinguishing characteristics do not provide sufficient entropy to attain a significant amount of cryptographic protection.

5.5 Preliminary Techniques Comparison

Key establishment techniques must be compared on the basis of their ability to satisfy distributed SAFEMITS network functionality and security requirements, while efficiently overcoming battlefield constraints. Table 38 highlights the benefits and deficiencies of the techniques examined in this report.

Technique	Type of Authentication	Type of Key Computation	Arbitra-ted?	Benefits	Deficiencies
Network-wide pre-deployed	Secret-key	Secret-key	No	Simple, energy-efficient	All data disclosed w/ single compromise
Node-specific pre-deployed	Secret-key	Secret-key	No	Simple, energy-efficient, granular keys	Limited to small groups, inflexible
J-Secure pre-deployed	Secret-key	Secret-key	No	Energy-efficient, granular keys	Limited by group size vs. number of colluding compromised nodes
Key distribution center-based	Secret-key	Secret-key	Yes	Energy-efficient, may have granular keys	Vulnerable to KDC compromise or many keys needed, inflexible
Symmetric key certificates	Secret-key	Secret-key	Yes	Energy-efficient, may have granular keys	Vulnerable to KTC compromise or many certificates needed
Identity-based symmetric	Secret-key	Secret-key	Yes	Energy-efficient, granular keys	Can't support unanticipated network merges or growth
Logical key hierarchy	Secret-key or public key	Secret-key	Yes/No	Computationally-efficient re-keying of large groups	Only useful for very large single hop groups
One-way function trees	Secret-key or public key	Secret-key	Yes/No	Computationally-efficient re-keying of large groups	Only useful for very large single hop groups
Rich Uncle	Public key / secret-key	Public key/ secret-key	Yes	Energy-efficient near super nodes, granular keys	Not energy-efficient more than two hops from super nodes unless symmetric key extensions are used
Pairwise w/ signature	Public key	Public key	No	Simple, granular keys	Not energy-efficient
Pairwise w/ MAC	Secret-key	Public key	No	Simple, granular keys, more energy-efficient than Pairwise w/ sign	Not energy-efficient, active attacks possible with single auth. key compromise
"Elected" Simple Key Distribution Center	Public key / secret-key	Public-key	No	Simple, granular keys, best average energy-efficiency for small groups	Group leader consumes more energy than other members
Cliques Group Diffie-Hellman w/ signature	Public key	Public key	No	Granular keys, more energy-efficient than Pairwise for groups	Only more energy-efficient in groups of six or more
Group Diffie-Hellman w/ MAC	Secret-key	Public key	No	Granular keys, more energy-efficient than GDH w/ sign	Active attacks possible with single auth. key compromise
Burmester-Desmedt w/ signature	Public key	Public key	No	Granular keys, more energy-efficient than GDH for multicast	Only more energy-efficient in multicast groups
Burmester-Desmedt w/ MAC	Secret-key	Public key	No	Granular keys, more energy-efficient than BD w/ sign	Active attacks possible with single auth. key compromise
Just-Vaudenary	Public key	Public key	No	Granular keys, more energy-efficient than BD	Consumes more energy than ESKDC
Attribute-based	Secret-key	Secret-key	No	Granular keys, energy-efficient	Does not provide strong cryptographic protection

Table 38 - Comparison of Key Establishment Techniques

The use of special nodes by certain protocol limits their flexibility. They are not useful during initial network self-configuration. These protocols pay a high energy-consumption price if large numbers of bits are transmitted to and from the special node(s).

From a security perspective, granular keys are desirable for the authentication of key exchange information during the keying protocol. Similarly, the application data keys established should have limited scope. The latter goal is more important than the former, however, since compromise of an authentication key can only be exploited by an active adversary, whereas compromise of a data encryption key can be exploited by a passive adversary.

Although secret-key algorithms are more energy-efficient than public key algorithms, the scope of keys used in most secret-key-based protocols is relatively large, more readily exposing the network to compromise. Exceptions to this key scope issue include the node-specific and identity-based symmetric keying protocols. Although they both have scalability and flexibility limitations, their energy-efficiency and use of granular keys make these protocols attractive for small SAFEMITS networks.

Public key algorithm-based keying protocols generally provide greater security than secret-key-based protocols by limiting the scope of generated keys. However, the amount of energy consumed by public key-based protocols, both through communications and computation, is of great concern.

The generally least energy-efficient, but most secure public key-based keying protocol is pairwise key establishment. Pairwise is relatively inefficient, but most secure, due to the fact that it establishes keys between only two SAFEMITS nodes. As the number of nodes that need to establish keys increases, the number of pairwise keys needed increases by the *square* of the number of nodes. Due to multi-hop routing, most nodes will not need to directly communicate with a large number of other nodes. However, even small groups such as six nodes can benefit greatly from some form of group keying.

Using secret-key-based authentication of the exchanged key management information, rather than public key certificate-based authentication, can reduce pairwise keying energy consumption. Although using a network-wide secret-key for key management authentication raises security concerns, the risk/reward benefits are acceptable for many scenarios. The average SAFEMITS node energy benefit of using secret-key-based authentication is shown in Table 39.

Processor	Pairwise RSA Average SAFEMITS Node Energy Consumption (mJ)		Percent Reduction in Energy for Secret-key
	Public Key Signature	Secret-key-based MAC	
MIPS R4000	132	123	6.8
SA-1110 "StrongARM"	130	122	6.1
Z-180	4200	2100	50
MC68328 "DragonBall"	1040	580	44
MCF5204 "ColdFire"	970	540	44
MMC2001"M-Core"	270	190	30
ARC 3 [29]	115	114	0.9

Table 39 - Benefits of Secret-key-Based MAC Authentication for Pairwise RSA[30]

An alternative method for establishing keys between two nodes is through use of an energy-endowed super node. The Rich Uncle protocol uses the asymmetric energy consumption characteristic of

[29] Simulation results.

[30] Does not include optional third pass for Pairwise RSA.

RSA to reduce the energy consumed by the SAFEMITS nodes at the expense of a super node. The benefit of using the Rich Uncle protocol over Pairwise RSA with signature authentication is shown in Table 40.

Node Composition	Average SAFEMITS Node Energy Consumption (mJ)		Energy Ratio, Pairwise/Rich Uncle
	Pairwise RSA, Signature	Rich Uncle, Pairwise, One-hop	
R4000 Processor w/ WINS Communications	132	96	1.89
R4000 Processor w/ MuRF Communications	49	27	1.81
Dragonball Processor w/ WINS Communications	620	179	3.45
Dragonball Processor w/ MuRF Communications	530	110	4.88

Table 40 - Comparison of Pairwise RSA[31] and Rich Uncle Energy Consumption

If a group of SAFEMITS nodes are connected via a single hop, group keying protocols such as Group Diffie-Hellman and Burmester-Desmedt without public key authentication may reduce energy consumption as shown in Table 41.

Node Composition	Average SAFEMITS Node Energy Consumption per Pairwise Keying Relationship[32] (mJ)		
	Pairwise RSA, MAC	Group Diffie-Hellman	Burmester-Desmedt
R4000 Processor w/ WINS Communications	123	72	110
Dragonball Processor w/ WINS Communications	580	2700	2100

Table 41 - Comparison of Pairwise RSA and Group Keying using Unicast Messages

If a multicast messaging is available within the SAFEMITS network, Table 42 shows that the benefits of using the Burmester-Desmedt group keying protocol without public key authentication become even greater.

[31] Does not include optional third pass for Pairwise RSA.

[32] Group Diffie-Hellman and Burmester-Desmedt computations are based on using six node groups.

Node Composition	Average SAFEMITS Node Energy Consumption per Pairwise Keying Relationship (mJ)		
	Pairwise RSA, MAC	Group Diffie-Hellman	Burmester-Desmedt
R4000 Processor w/ WINS Communications	123	79	52
Dragonball Processor w/ WINS Communications	580	2700	2000

Table 42 - Comparison of Pairwise RSA and Group Keying using Multicast Messages

The various key management protocols described in Section 5 can be compared over many different dimensions, for various scenarios. However, a more useful methodology for further study is to evaluate the performance of these protocols over an entire distributed SAFEMITS network simulated using real-world mission and environmental parameters.

6 Network-wide Approaches

The first phase of our research has analyzed and compared various key establishment protocols for use in distributed SAFEMITS networks. Our results have shown that a single keying protocol will likely not be optimal in all scenarios, nor for the entire duration of a distributed SAFEMITS network mission. The second phase of our research has begun to simulate and analyze approaches to efficiently use the various keying protocols in a coordinated manner throughout the SAFEMITS network.

We discuss our recent research in the context of the following areas:

- the Sanders simulation upon which our simulation is based,
- self-organization steps in a distributed SAFEMITS network,
- group determination,
- hybrid keying approaches, and
- energy-aware keying approaches.

6.1 SAFEMITS Network Simulation

Our simulation of keying protocols within a distributed SAFEMITS network is dependent on, and tightly integrated with, a distributed SAFEMITS network simulation developed by Dr. Diane Mills and Melissa Chevalier of Sanders, a Lockheed Martin Company. The Sanders MATLAB-based simulation provides the capability to perform several fixed or randomized SAFEMITS field laydowns for a wide variety of different parameters including topology determination, link cost determination, communications range, SAFEMITS range, and number of nodes.

SUNYIT research team has added several capabilities to the Sanders simulation including the ability to perform candidate hybrid and energy-aware keying protocols over the SAFEMITS array for various topologies, link costs, and communications ranges. These simulation additions generate both energy consumption data and figures that display pairwise and group node interconnections. The SUNYIT research team simulation additions were used to generate the data and figures that are shown and analyzed in the following sections.

Although we tested our simulation implementation on randomized SAFEMITS field laydowns, for consistency and relevancy reasons we present our figures and analysis based on the Group A SAFEMITS positions shown in Table 4.1.1-1 of the SAFEMITS August '00 Test Plan Version 1.0. Figure shows these positions and corresponding node numbers graphically:

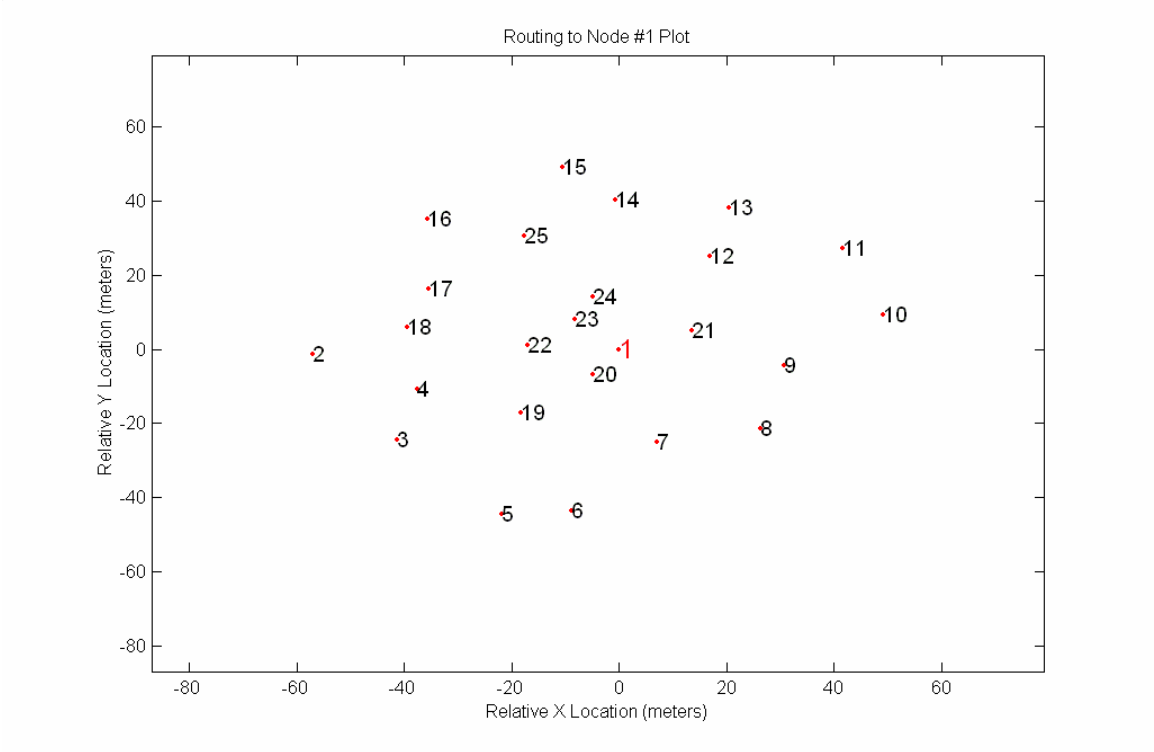

Figure 17 - SAFEMITS August '03 Planned Group A SAFEMITS Positions

6.2 Integrating Key Establishment with Network Self-Organization

While integrating our keying protocols with the steps of SAFEMITS network self-organization, we encountered an issue in determining what stage of self-organization is best to perform key establishment. Without security, SAFEMITS nodes self-organize by first discovering their neighbors, and then performing some protocol to determine the routing paths to destinations such as gateways. Only after routing paths have been established may application messages be exchanged.

When security is added to a SAFEMITS network, key establishment must be performed prior to the exchange of application messages in order to protect these messages. The issue we have identified is whether key establishment requires information from the routing path determination protocol, and whether the routing protocol requires cryptographic protection furnished by security services supported by key establishment. If key establishment is dependent upon or occurs after routing, then key establishment must also be performed upon each re-routing. Conversely, if key establishment occurs before routing, re-routing may occur independently of key establishment.

Whether to perform key establishment before or after routing determination may be influenced by SAFEMITS node density. In a dense SAFEMITS network, routing protocols will be less constrained in choosing routing paths. With fewer constraints, the network will be able to optimize and re-optimize for a variety of different factors such as remaining battery energy. With more options, a dense network is more amenable to re-routing. Thus, performing key establishment once before routing avoids the cost of rekeying after each re-routing.

A dense SAFEMITS network is also more able to use energy-saving group keying protocols. Group keying protocols such as Group Diffie-Hellman and Burmester-Desmedt require groups of at least five singly-hop-connected nodes before their efficiency gains surpass those of simple pairwise keying. Dense SAFEMITS networks are more likely to have the required number of singly-hop-connected nodes to allow group keying protocol efficiencies to be garnered.

A sparse SAFEMITS network is more constrained in choosing routing paths than a dense one, and thus less likely to perform re-routing. Since re-routing will occur less often, and group key protocols will likely not be advantageous, performing a simple pairwise scheme between only routing path nodes may be more energy-efficient. Therefore, a sparse SAFEMITS network may more efficiently establish keys after routing determination than before.

A multi-hop-connected keying protocol requires that routing occur before key establishment. We have currently focused on singly-hop-connected protocols since communications energy consumption is often our greatest constraint, but some scenarios may benefit from multi-hop-connected keying.

However, if the routing determination protocol itself requires cryptographic protection, some type of key establishment must occur prior to routing. We expect routing protocols will require confidentiality, integrity, and/or authentication protection, and will therefore require some keys establishes apriori.

Furthermore, there may be a benefit to integrating key establishment protocols directly with routing determination protocols. Key establishment protocols often have a few rounds, the initial rounds of which may be amenable to inclusion with portions of the routing determination protocol. At this time we recommend against integration, since the wide variety of key establishment and routing determination protocols causes the number of integration possibilities to grow by the product of the number of keying and routing protocols. As noted later, however, this is an interesting area for additional research.

In this document we have chosen to examine key establishment prior to routing. Although there is a need to examine the possibility of performing key establishment during or after routing, we suggest this be deferred to future research.

6.3 Group Determination

The first step of a network-wide key establishment approach is determination of any and all keying groups. Following the discovery step of SAFEMITS network self-organization, the key establishment phase will identify all singly-hop-connected groups, all star groups, and potentially multi-hop-connected groups.

A singly-hop-connected group is defined as a group of SAFEMITS nodes that can each transmit and receive to every other SAFEMITS node in the group. Since the Group Diffie-Hellman and Burmester-Desmedt keying protocols require all group members to be interconnected, our simulation required these protocols to use singly-hop-connected groups. Our algorithm for finding singly-hop-connected groups requires groups to have at least five members and not have more than one member in common with another singly-hop-connected group. For the SAFEMITS August '00 Group A SAFEMITS positions, three singly-hop-connected groups are created when the maximum SAFEMITS node communications range was set to 60 meters as shown in Figure . The three groups are of sizes 5, 8, and 13 with nodes 1, 8, and 20 members of two groups each. Links that are pairwise connected only are shown via red lines.

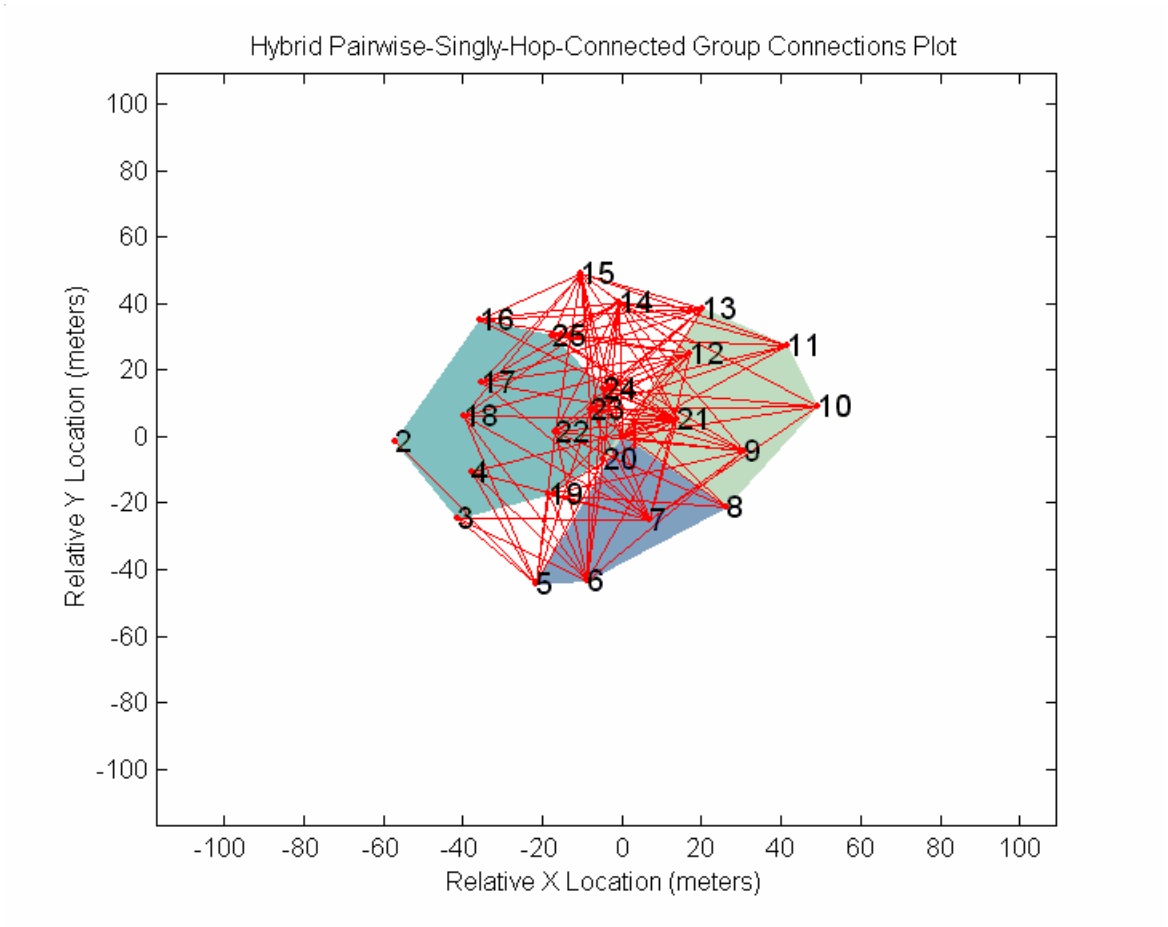

Figure 18 - Singly-Hop-Connected Groups with Communications Range of 60 Meters

We define a star group as a group of SAFEMITS nodes that can each transmit and receive to a single "leader" node within the group. The Simple Key Distribution Center protocol requires a star group. Since nodes in a star group need only all connect to a single node, unlike singly-hop-connected groups that must interconnect each and every node, star groups will generally be larger than singly-hop-connected groups for the same SAFEMITS field laydown. Using the SAFEMITS August '00 Group A SAFEMITS positions as an example, the entire 25-node SAFEMITS field forms a star group when the maximum SAFEMITS node communications range is 60 meters as shown in Figure .

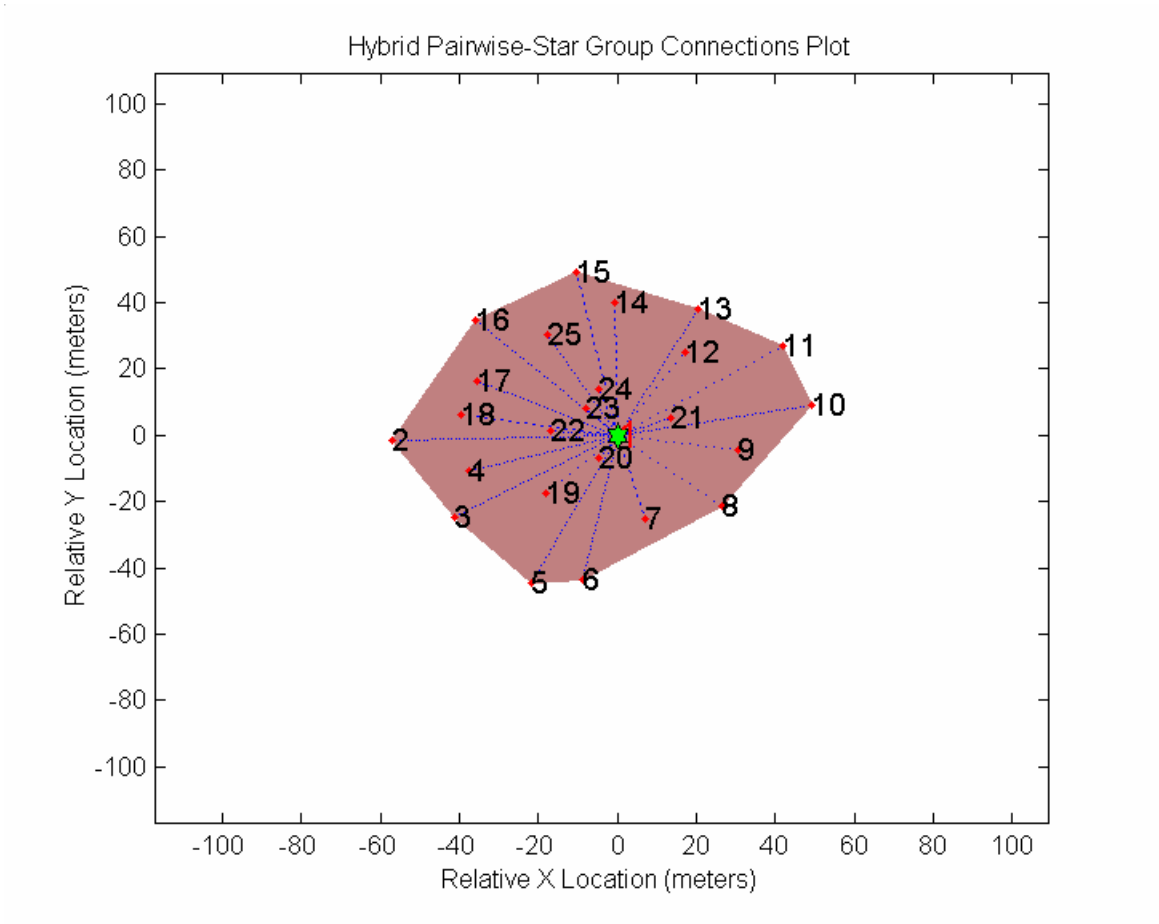

Figure 19 - Star Group with Communications Range of 60 Meters

A multi-hop-connected group is defined as a group of SAFEMITS nodes that are interconnected via two or more hops. Provided nodes are not isolated, multi-hop-connected groups may always be formed. However, some type of routing must be performed prior to multi-hop-connected group formation. Multi-hop-connected groups are generally not attractive for key establishment due to the large amount of communications energy they consume. Although there may potentially be some application of multi-hop-connected groups for key establishment, we have chosen to defer such investigation.

6.4 Hybrid Approaches

Since a single keying protocol will not be optimal for all nodes and all scenarios, SAFEMITS nodes must be capable of performing multiple keying protocols. We have analyzed various approaches to using multiple keying protocols in a single distributed SAFEMITS network.

6.4.1 Pairwise and Group Diffie-Hellman Hybrids

We examined establishing keys within a distributed SAFEMITS network using a combination of the Pairwise and Group Diffie-Hellman protocols. Our approach was to first find all of the singly-hop-connected groups within the SAFEMITS network field. Next, we significantly pruned this list by finding the largest, the next largest, etc., provided each successively smaller group had at least five members and did not have more than one member in common with a previous group. The five-

member minimum is necessary since groups with less than five members are more efficiently keyed using fully interconnected pairwise rather than Group Diffie-Hellman. The overlapping restriction prevents inefficiencies of forming two groups that have almost entirely identical membership. Although our approach is likely overly conservative of forming more groups and thus sub-optimal, we expect the resulting energy consumption values to be close enough to optimal for our simulation purposes.

We examined two GDH techniques that we call *Naïve GDH* and *GDH*. Naïve GDH performs the GDH keying protocol on each group, assigning roles arbitrarily based on ascending node number. GDH performs the GDH keying protocol in a more intelligent fashion, assigning the two communications-intensive roles to the two nodes that are closest, in a communications range^4 fashion, to the other group nodes. Both GDH simulation implementations use the IKA.3 protocol scheme.

Furthermore, we examined these two GDH techniques for two different types of RF transmit power control. First, we examined the energy consumption under the assumption that each node knows exactly the right amount of radiated power it must transmit to be received by the receiver with the desired minimum bit-error-rate. A comparison of Naïve GDH, GDH, and pairwise only for various communications ranges and per transmission control is shown in Table 43.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)			Singly-hop-connected Group Sizes		
	Pairwise Only	Pairwise-'Naive' GDH Hybrid	Pairwise-GDH Hybrid			
30	.55	.58	.58	5	-	-
35	.82	.87	.87	6	5	-
40	1.26	1.22	1.21	8	6	5
45	1.73	1.65	1.62	9	6	5
50	2.28	2.03	1.95	9	8	6
60	4.42	3.54	3.39	13	8	5
70	6.63	4.93	4.78	15	7	6
80	9.67	6.34	6.11	18	6	-
90	13.42	7.07	6.53	21	5	-

Table 43 – Pairwise-GDH Energy Consumption, Per Transmission Power Control

Although a benefit of optimizing GDH roles is apparent from the fact that the energy consumption values for GDH are lower than Naïve GDH, the differences are relatively small. Also apparent from Table 43 is the fact that increasing communications range results in increased energy consumption per node. The energy consumption increase is due to at least two factors: more nodes are now connected to one another, and the distance between additional nodes is greater and requires more RF transmission power.

The benefit of using a pairwise-GDH hybrid over pairwise-only appears to be realized when the average group size is greater than six. This threshold group size is achieved at a communications range of 40 meters in the scenario we analyzed in this simulation. Further group size increases provide additional reductions of energy consumption.

We examined this same scenario in Table 44 with the exception of fixing the transmit power to the maximum communications range at all times. Fixing the transmit power to the maximum increases the energy consumption for all trials, but also further increases the benefit of using a pairwise-GDH hybrid over pairwise-only.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)			Singly-hop-connected Group Sizes		
	Pairwise-Only	Pairwise-'Naive' GDH Hybrid	Pairwise-GDH Hybrid			
30	.67	.70	.70	5	-	-
35	1.14	1.15	1.15	6	5	-
40	1.98	1.80	1.80	8	6	5
45	3.21	2.78	2.78	9	6	5
50	4.95	3.98	3.98	9	8	6
60	11.80	8.19	8.19	13	8	5
70	23.27	14.84	14.84	15	7	6
80	42.24	23.13	23.13	18	6	-
90	70.80	28.96	28.96	21	5	-

Table 44 – Pairwise-GDH Energy Consumption, No Transmit Power Control

6.4.2 Pairwise and Burmester-Desmedt Hybrids

We examined establishing keys within a distributed SAFEMITS network using a combination of the Pairwise and Burmester-Desmedt protocols. As with the pairwise-GDH hybrid, we found a pruned list of singly-hop-connected groups with at least five members and no more than one member node in common. To highlight the potential advantages of Burmester-Desmedt, we simulated the exchange of key management information via multicast communication. Per transmission RF power control for multicast was simulated as the amount of radiated RF power required to communicate with all of the recipients of a multicast transmission at the minimum bit-error-rate. The key management energy consumption results of our simulation are shown in Table 45.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)		Singly-hop-connected Group Sizes		
	Pairwise-Only	Pairwise-BD Hybrid			
30	.55	.42	5	-	-
35	.82	.59	6	5	-
40	1.26	.79	8	6	5
45	1.73	1.05	9	6	5
50	2.28	1.30	9	8	6
60	4.42	2.28	13	8	5
70	6.63	3.32	15	7	6
80	9.67	4.29	18	6	-
90	13.42	5.33	21	5	-

Table 45 – Pairwise-BD Energy Consumption, Per Transmission Power Control

The pairwise-BD hybrid is more energy-efficient than pairwise-only for all simulated communications ranges, and even in the 30 Meter communications range trial where only a single five member group is created within the 25-node SAFEMITS network.

When the RF transmission power control is fixed to maximum, the benefits of the pairwise-BD hybrid, as shown in Table 46, become even more dramatic. The energy consumption reduction

provided by the pairwise-BD hybrid ranges from about 31% at the 30 Meter communications range to a whopping 85% at the 90 Meter communications range.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)		Singly-hop-connected Group Sizes		
	Pairwise-Only	Pairwise-BD Hybrid			
30	.67	.46	5	-	-
35	1.14	.66	6	5	-
40	1.98	.93	8	6	5
45	3.21	1.35	9	6	5
50	4.95	1.80	9	8	6
60	11.80	3.38	13	8	5
70	23.27	5.76	15	7	6
80	42.24	8.60	18	6	-
90	70.80	10.65	21	5	-

Table 46 – Pairwise-BD Energy Consumption, No Transmit Power Control

6.4.3 Pairwise and Simple Key Distribution Center Hybrids

We examined establishing keys within a distributed SAFEMITS network using a combination of the Pairwise and Simple Key Distribution Center (SKDC) protocols. Unlike the pairwise-GDH and pairwise-BD hybrids, we developed a pruned list of star groups with at least six members and no more than two member nodes in common. Despite the more restrictive group size requirements, star group sizes were generally larger, thus accentuating the benefit of the SKDC protocol. For the SAFEMITS network scenario we simulated, the entire 25-node field is a member of a single star group for communications ranges of 60 meters and greater. The key management energy consumption results of our pairwise-SKDC simulation are shown in Table 47.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)		Star Group Sizes		
	Pairwise Only	Pairwise-SKDC Hybrid			
30	.55	.36	10	8	6
35	.82	.52	13	-	-
40	1.26	.69	16	-	-
45	1.73	.77	17	-	-
50	2.28	.69	22	-	-
60	4.42	.50	25	-	-
70	6.63	.50	25	-	-
80	9.67	.50	25	-	-
90	13.42	.50	25	-	-

Table 47 – Pairwise-SKDC Energy Consumption, Per Transmission Power Control

When the RF transmission power control is fixed to the maximum, the benefits of SKDC as shown in Table 48 remain significant, although not as great as in the per transmission control trials shown in Table 47.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)		Star Group Sizes		
	Pairwise Only	Pairwise-SKDC Hybrid			
30	.67	.45	10	8	6
35	1.14	.71	13	-	-
40	1.98	1.05	16	-	-
45	3.21	1.46	17	-	-
50	4.95	1.44	22	-	-
60	11.80	1.58	25	-	-
70	23.27	2.77	25	-	-
80	42.24	4.59	25	-	-
90	70.80	7.24	25	-	-

Table 48 – Pairwise-SKDC Energy Consumption, No Transmit Power Control

We note that in this version of the protocol, we do not provide perfect forward secrecy, unlike the GDH and BD versions described in this section. If perfect forward secrecy is required, SKDC could be modified to provide this service, at a cost to energy efficiency. Although perfect forward secrecy may not be required for secrecy of routing information, it would likely be required for the confidentiality protection of application-layer messages.

6.4.4 Comparison of Approaches

We compare the three hybrid approaches Pairwise-GDH, Pairwise-BD, and Pairwise-SKDC against the conventional pairwise-only scheme. Our analysis examines both average and maximum energy consumption of each scheme. We also examine the effect of node density on group size.

6.4.4.1 Average Per Node Energy Consumption

Table X49 compares the average per node key management energy consumption for the pairwise-only baseline with three dual-protocol hybrid schemes when the radiated RF transmission power control is controllable on a per transmission basis. For the majority of communications ranges, the dual-protocol hybrid schemes are significantly more energy-efficient than pairwise-only, with the Pairwise-SKDC hybrid being most efficient.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)			
	Pairwise Only	Pairwise-GDH Hybrid	Pairwise-BD Hybrid	Pairwise-SKDC Hybrid
30	.55	*.58*	.42	**.36**
35	.82	*.87*	.59	**.52**
40	*1.26*	1.21	.79	**.69**
45	*1.73*	1.62	1.05	**.77**
50	*2.28*	1.95	1.30	**.69**
60	*4.42*	3.39	2.28	**.50**
70	*6.63*	4.78	3.32	**.50**
80	*9.67*	6.11	4.29	**.50**
90	*13.42*	6.53	5.33	**.50**

Table 49 - Hybrid Keying Average Energy Consumption, Per Transmission Power Control

As shown in Table X50, when the SAFEMITS node's RF transmission power is not controllable, the Pairwise-BD hybrid is most energy-efficient at lower communications ranges, whereas the Pairwise-SKDC hybrid is most energy-efficient at greater communications ranges. However, the Pairwise-BD hybrid benefit only occurs when multicast transmission is available, thus demonstrating the importance of this capability to key management energy efficiency.

Communications Range (meters)	Key Management Energy Consumption (Joules / node)			
	Pairwise Only	Pairwise-GDH Hybrid	Pairwise-BD Hybrid	Pairwise-SKDC Hybrid
30	.67	*.70*	.46	**.45**
35	1.14	*1.15*	**.66**	.71
40	*1.98*	1.80	**.93**	1.05
45	*3.21*	2.78	**1.35**	1.46
50	*4.95*	3.98	1.80	**1.44**
60	*11.80*	8.19	3.38	**1.58**
70	*23.27*	14.84	5.76	**2.77**
80	*42.24*	23.13	8.60	**4.59**
90	*70.80*	28.96	10.65	**7.24**

Table 50 - Hybrid Keying Average Energy Consumption, No Transmit Power Control

6.4.4.2 Maximum Energy Consumption

Table X51 compares the maximum key management energy consumption for the pairwise-only baseline with three dual-protocol hybrid schemes when the radiated RF transmission power control is controllable on a per transmission basis. For the majority of communications ranges, the pairwise-BD hybrid protocol is most efficient due to its symmetric nature. Conversely, the asymmetric nature of the pairwise-SKDC hybrid protocol results in the largest maximum energy consumption for most of the evaluated communications ranges.

Communications Range (meters)	Maximum Key Management Energy Consumption (Joules)			
	Pairwise Only	Pairwise-GDH Hybrid	Pairwise-BD Hybrid	Pairwise-SKDC Hybrid
30	1.19	1.49	**1.06**	*1.63*
35	1.90	1.84	**1.34**	*2.30*
40	2.52	2.37	**1.70**	*3.38*
45	3.36	3.13	**1.69**	*4.19*
50	4.79	4.27	**2.45**	*6.41*
60	6.66	5.66	**2.92**	*9.42*
70	8.29	8.13	**5.60**	*9.42*
80	*14.60*	12.47	**8.30**	9.42
90	*23.98*	23.14	13.76	**9.42**

Table 51 – Maximum Energy Consumption, Per Transmission Power Control

Table 52 shows that when the RF transmission power is not controllable per transmission, the advantage of a symmetric protocol such as pairwise-BD is more striking. Similarly, the disadvantage of using a pairwise-SKDC protocol where a single node incurs much of the energy consumption is also shown.

Communications Range (meters)	Maximum Key Management Energy Consumption (Joules)			
	Pairwise Only	Pairwise-GDH Hybrid	Pairwise-BD Hybrid	Pairwise-SKDC Hybrid
30	1.41	1.62	**1.08**	*1.89*
35	2.43	2.15	**1.34**	*3.08*
40	3.66	3.41	**1.74**	*4.92*
45	5.65	5.27	**1.87**	*7.33*
50	8.78	8.40	**2.58**	*12.09*
60	17.42	16.66	**5.67**	*24.63*
70	29.42	29.04	**12.58**	*42.44*
80	47.76	46.88	**19.45**	*69.76*
90	74.25	73.75	**26.98**	*109.50*

Table 52 - Maximum Energy Consumption, No Transmit Power Control

6.4.4.3 Node Density vs. Group Size vs. Group Type

Since group sizes have such a dramatic effect on hybrid keying benefits, we more closely examine the relationships between node density, group sizes, and group types. We use the term node density to describe the ratio between communications range and the average minimum per node neighbor distance. For the SAFEMITS node field laydown that was analyzed in our simulation, the average minimum per node neighbor distance was 14.69 meters. This SAFEMITS node field is fully connected when communications ranges are at least 25 meters or a node density of about 1.7.

The advantage of referring to node density instead of communications range is that node density is dimensionless and can be used in a wider variety of environments. For instance, if communications ranges are in the 100 meter range and the average

The plot of pairwise, singly-hop-connected group, and star group connections for a communications range of 30 meters and node density ratio of 2.04 is shown in Figure . Note, only one five-member

singly-hop-connected group is formed, whereas three star groups of sizes 10, 8, and 6 are formed. Consequently, the greater key management energy consumption reductions are found with the star-group-based SKDC scheme, rather than the singly-hop-connected-based GDH and BD schemes.

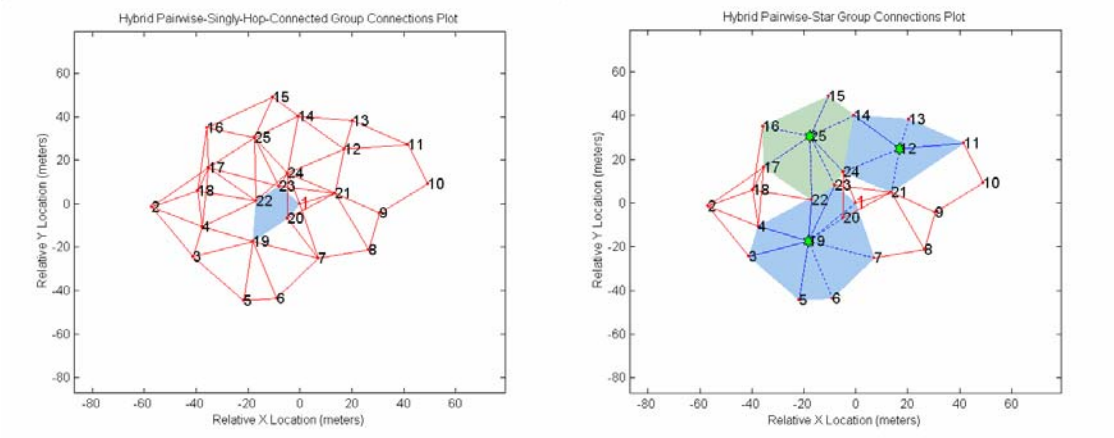

Figure 20 - Pairwise and Group Connections, Communications Range = 30 Meters

Increasing the communications range to 35 meters and the node density to 2.38 results in an additional six node singly-hop-connected group being formed as shown in Figure . Also, a single 13-node star group is formed at this node density.

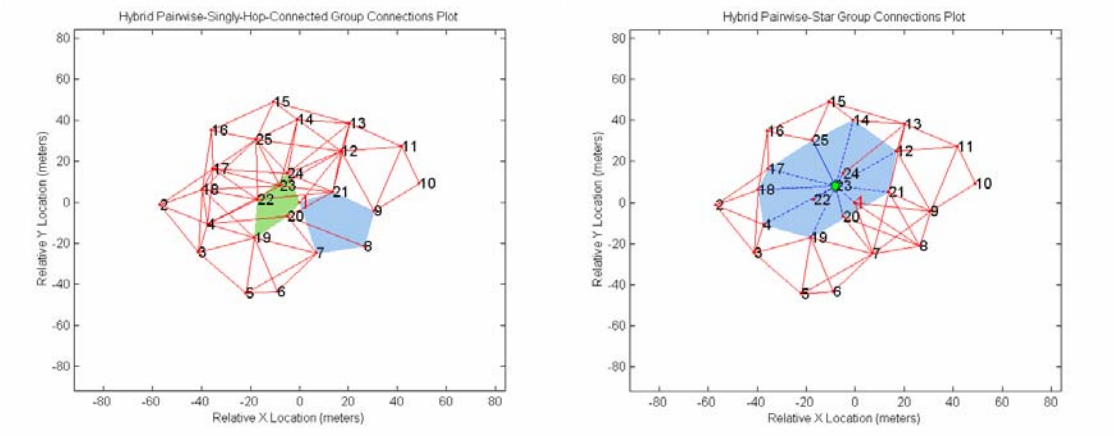

Figure 21 - Pairwise and Group Connections, Communications Range = 35 Meters

With a communications range of 40 meters, larger singly-hop-connected groups and a larger star group are formed within the SAFEMITS field as shown in Figure .

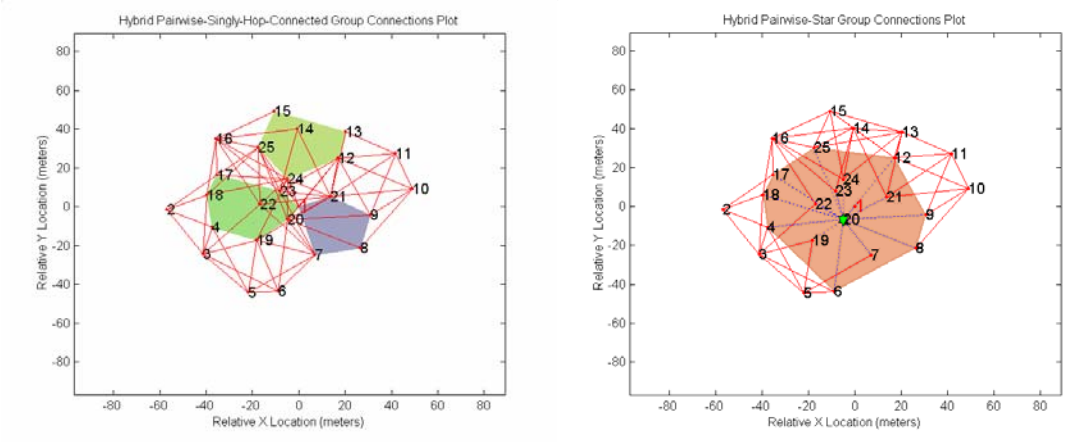

Figure 22 - Pairwise and Group Connections, Communications Range = 40 Meters

These groups become larger still when the communications range is increased to 45 meters as shown in Figure .

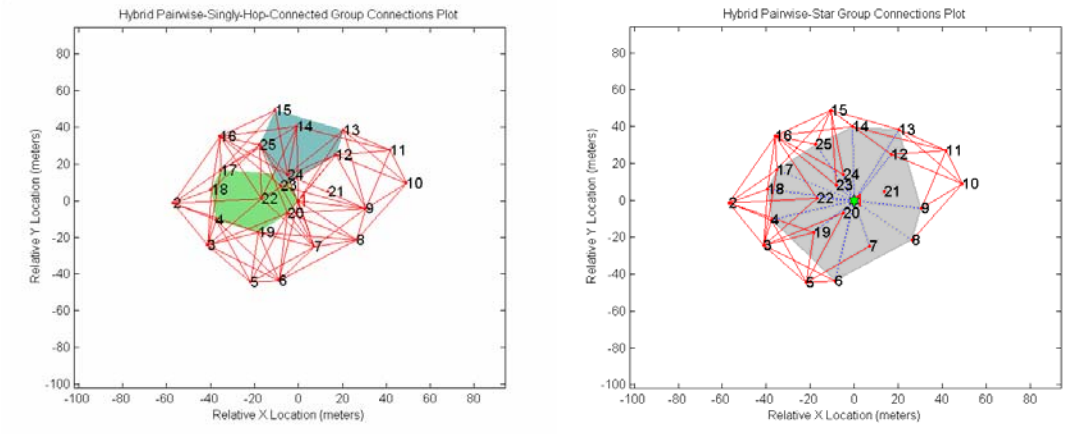

Figure 23 - Pairwise and Group Connections, Communications Range = 45 Meters

A parallel effect of the larger group sizes is the reduction of the number of pairwise connections that must take to interconnect the remaining links of the SAFEMITS node field. This is quite apparent in the Figure hybrid pairwise-star group connections plot where nodes 7 and 8 require only one additional pairwise connection each.

6.5 Energy-Aware Approaches

6.5.1 Key Protocol Roles

Some public key algorithms, such as RSA, provide a difference in computational energy consumed of as much as a factor of twenty. Such an asymmetric relationship can be exploited by an *energy-aware* network-wide approach that identifies nodes that need to conserve their battery energy. An energy-aware approach may identify energy-depleted nodes by exchanging *battery energy-remaining* values.

With more sophistication, protocols could be designed to identify nodes that will be energy-depleted a priori. Routing configurations, such as that shown in Figure , identify likely communications choke points, which are likely candidates for battery energy depletion.

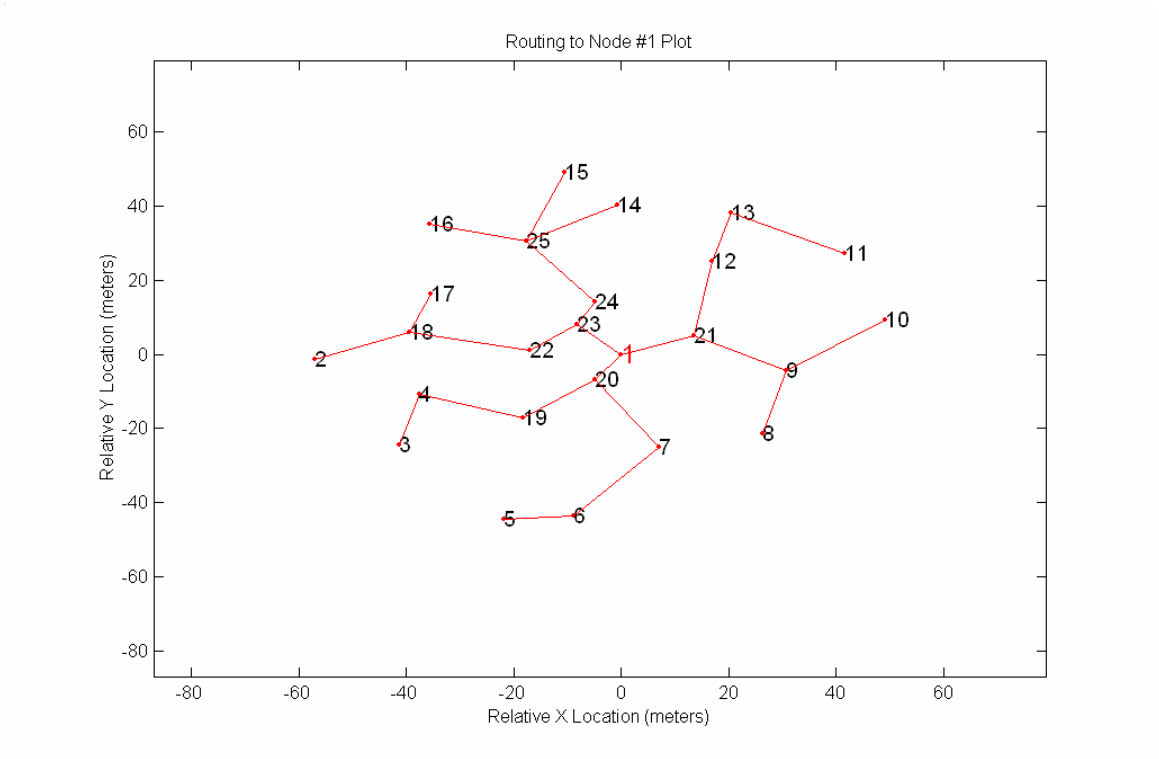

Figure 24 - Routing to Node 1 Plot

Similarly, group key management protocols such as Group Diffie-Hellman and SKDC offer role-dependent energy consumption. Energy-depleted (or to-be-depleted) nodes, such as Nodes 1, 20, 21, and 23 should not perform the "controller" role for group key management protocols.

6.5.2 Parasite Protocol

Yet another key management approach available to energy-depleted nodes is the Parasite protocol. As was earlier shown in the Rich Uncle protocol, nodes can use the asymmetric energy consumption characteristics of RSA to minimize their energy consumption. Rather than establish a keying relationship with just one other node as is done in the Rich Uncle protocol, a parasite would establish the keying relationship as a first step toward obtaining a nearby group's common key. By obtaining the group's key, the parasite has established a keying relationship with all of the group's nodes, without participating in the potentially expensive group key management protocol.

In

Figure , Node 25 is a potential benefactor from the parasite protocol. Any one of Nodes 22, 23, or 24 can establish a pairwise relationship with and send the group key to Node 25. By doing so, Node 25 shares common keys with each of Nodes 22, 23, and 24. Although the routing plot of Figure shows that Node 25 is only connected to Node 24, subsequent re-routing may require Node 25 to be connected to Nodes 22 or 23.

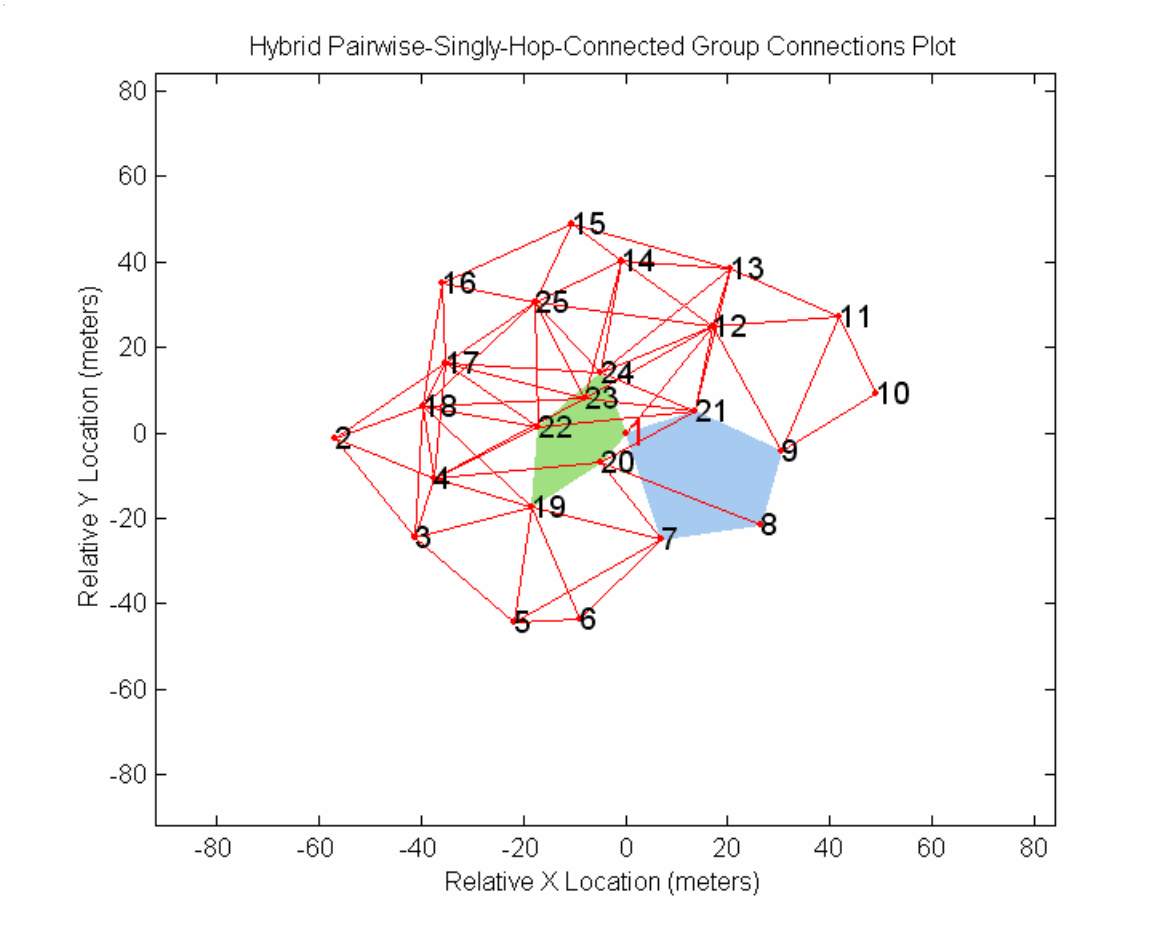

Figure 25 - Pairwise and Connections, Communications Range = 35 Meters

6.5.3 Time Varying Approaches

In addition to varying techniques based on locations within the network, key management techniques may also vary over time. Network-wide pre-deployed keys could potentially be used to support establishment of initial keying relationships, but using such keys throughout the lifetime of a SAFEMITS network increasingly risks compromise over time. Instead, a timely transition to more granular keys is advised. Similarly, since SAFEMITS nodes expend battery energy over time, trading off security for energy efficiency may also be warranted in the later stages of a SAFEMITS network's lifetime.

6.6 Specialization

In situations where the density of SAFEMITSs exceeds the communications requirements, some SAFEMITSs may "specialize" in certain roles. Since all SAFEMITSs can perform communications, communications, and security functions, it may be beneficial to the entire SAFEMITS network to have some SAFEMITSs perform mostly communications, others concentrate on communications, and still others concentrate on security functions.

Specifically for key management, a SAFEMITS node may self-elect to take on the energy-consumptive role of key distribution center or Rich Uncle, to spare its surrounding SAFEMITS nodes from expending a great deal of energy on security. A SAFEMITS node that self-elects to serve in the super node role of the Rich Uncle protocol we call a *Pseudo Rich Uncle.*[33]

[33] Ken Theriault of BBN Technologies suggested this concept in a conversation at the SAFEMITS IT Workshop on April 4, 2000.

7 Next Steps

Although our research has identified key management energy-efficiency improvements for a number of scenarios, further improvements are possible. We have identified the following areas were additional research would enhance key management performance:

- Development of an optimized group determination algorithm – The algorithm we are currently using is sub-optimal since it simply finds the largest group available, whereas a smaller group may provide a greater reduction in energy consumption depending on the relative positions of the group members. Furthermore, the optimal amount of overlapping between groups has not been determined.

- Finding GDH-optimized groups and optimizing member roles – Finding singly-hop-connected groups is an overly restrictive simplified approach towards performing hybrid keying using a Group Diffie-Hellman protocol. A stricter approach would not require all members to interconnect, but rather simply require the controllers to connect to all members and require all non-controlling nodes be connected to their protocol "next-door neighbors". Moreover, selection of member roles can be further optimized to minimize the communications energy by having protocol "next-door neighbors" to match with the actual physical "next-door neighbors".

- Multiple group keying protocols in a single hybrid protocol – Thus far, we have examined hybrid protocols that included a group keying protocol and a pairwise keying protocol. We believe there are scenarios, especially with much larger SAFEMITS networks, where two or more different group keying protocols in addition to a pairwise keying protocol may provide a better hybrid protocol than just one group keying protocol alone.

- Multi-hop keying – Although establishing keys via protocols that require multiple hops appears to be less energy efficient, we believe there may be scenarios in densely populated SAFEMITS networks where multi-hop keying may be effective.

- Parasite keying – We have qualitatively identified scenarios where Parasite keying is advantageous, but have not yet shown these benefits quantitatively. These benefits are best shown via simulation in the near-term, building upon our existing hybrid keying protocols.

- Per-node group determination and role determination algorithms – As we transition from research and simulation to integration and demonstration, it is important we appropriately transform our algorithms as well. Currently, our algorithms assume a degree of omnipotence regarding the locations of neighboring SAFEMITS nodes, the possible interconnections, the groups to be formed, and each node's given group role. Our algorithms must assume less to handle the asynchronous nature of self-assembling networks, including making determinations with limited information that may result in sub-optimal configurations.

- Integration of routing and keying protocols – Despite the additional complexity of integrating routing and key establishment protocols, there may be significant advantages in combining some aspects of these protocols. For instance, some key establishment protection is necessary to protect routing determination protocols. However, some multi-hop key establishment protocols require routing to already be determined. Integrating portions of both protocols together may provide energy reductions not possible with these functions separated.

- Further protocol exploration – As we develop increasingly more sophisticated simulations and development demonstrations, new issues with the protocols will become important.

Different communication channel models will have varying impact on the latency of different cryptographic protocols and on the ability of the network to run multiple protocols concurrently. The effect of SAFEMITS node dozing on different keying protocols must be examined and deficiencies addressed. Asymmetric communication links between nodes seriously impact the use of certain key management protocols. Further development of amortization techniques and simulating / testing is needed in order to minimize energy consumption and latency.

- Key management during *post routing-establishment* and network re-seeding phases – Once the *routing infrastructure* has been established SAFEMITS nodes can utilize (at a cost) remote resources. During network re-seeding (or during significant network disruptions) the network may consist of regions that for a moment completely lack routing, regions that have well established routing, and mixed regions with only partial, highly irregular routing in place. The chaotic nature (and potentially low latency tolerance) of such situations will be especially challenging to energy efficient key establishment. The joining of two SAFEMITS networks also presents similar challenges to key management. Both of these phases will require different key management protocol mixes than the mixes used during the *pre-routing* and *routing establishment* phases

- Port simulation from MATLAB to 'C' – MATLAB is an excellent platform for simulation and research that can easily generate useful figures and graphics, but it is not easily integrated into a prototype development. As we transition towards developing a security implementation to validate and progress our research, we should first take the intermediate step of porting our MATLAB-based algorithms to the 'C' programming language. Not only will porting to 'C' allow us to more easily develop on prototype SAFEMITS nodes later, the improved performance of 'C' allows us to simulate larger SAFEMITS networks to determine how well our approaches scale.

- Researching other security services – Key management is but one of the many security services that must be supported by the distributed SAFEMITS network. Our research should additionally examine other security services, such as integrity, authentication, and non-repudiation, to determine efficient and secure methods of providing these services.

- Implementation of security services – Finally, we must validate our research via SAFEMITS prototype-based experiments. The challenges of implementation and the many real world issues such as energy, latency, and network self-assembly provide an excellent environment for identifying the critical elements of the research. In addition to the key management, a prototype implementation must include basic security services such as confidentiality, integrity, and authentication. An independent red-team security analysis of our design and implementation will also provide great value to this security research.

References[34]

[Aether95] Aether Wire Location, Corp., "Low-Power, Miniature, Distributed Position Location and Communication Devices Using Ultra-Wideband, Nonsinusoidal Communication Technology", AEther Wire Location, Corp., Semi-Annual Technical Report, ARPA Contract J-FBI-94-058, July 1995.

[Agre99] Agre, J., L. Clare, G. Pottie and N. Romanov, "Development Platform for Self-Organizing Wireless SAFEMITS Networks", Proceedings of SPIE AeroSense '99.

[Agre00a] Agre, J., et al, "Communications Positioning Integrated Network (SPIN): Providing Situational Awareness to the Warfighter", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Agre00b] Agre, J. and L. Clare, "An Integrated Architecture for Cooperative Communications Networks", IEEE Computer, May 2000.

[Anderson97] Anderson, R. and M. Kuhn, "Low Cost Attacks on Tamper Resistant Devices", Security Protocols 5th Annual Workshop, Paris, France, 7-9 April, 1997.

[Aoki00] Aoki, K., and Lipmaa, H., "Fast Implementations of AES Candidates", Proceedings of the Third AES Candidate Conference, 13-14 April 2000, New York, New York.

[Ateniese99] Ateniese, G., M. Steiner and G. Tsudik, "New Multi-party Authentication Services and Key Agreement Protocols", in submission (February 5, 1999), 19 pages.

[Balenson00] Balenson, D., D. Branstad, D. Carman, P. Kruus and B. Matt, "Key Management for Distributed SAFEMITS Networking", Brief presented at the DARPA SAFEMITS IT Workshop, April 4-5, 2000.

[Barrett98] Barrett, M., M. Little, A. Poylisher, M. Gaughan and A. Tardif, "Intelligent Agents for Vulnerability Assessment of Computer Networks", Proceedings of the ARL Federated Laboratory 2nd Annual Symposium, 1998.

[Bellare93] Bellare, M. and P. Rogaway, "Entity Authentication and Key Distribution, "in *Advances in Cryptology: Proceedings of Crypto93*, LNCS 773, Springer-Verlag (1993), 232-249.

[Beller93] Beller, M. and Y. Yacobi, "Fully-fledged two-way public key authentication and key agreement for low cost terminals", Electronics Letters, 29 (May 27, 1993), 999-1001.

[Bosselaers93] Bosselaers, A., R. Govaerts, and J. Vandewalle, "Comparison of three modular reduction functions", Katholieke Universiteit Leuven, Dept. Electrical Engineering-ESAT, 25 October 1993.

[Boyd93] Boyd, C. and W. Mao, "On a Limitation of BAN Logic," Advances in Cryptology: Proceedings of – *EUROCRYPT'93*, LNCS 765, Springer-Verlag (1994), 240-247.

[Blundo92] Blundo, C., A. de Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Advances in Cryptology: Proceedings of Crypto92*, E. F. Brickell, ed., LNCS 740, Springer-Verlag (1992), 471–486.

[Burmester94] Burmester, M. and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System", in *Advances in Cryptology – EUROCRYPT'94* (May 1994), 275-286.

[34] Not all references are cited in this report.

[Canetti97] Canetti, R., "Toward Realizing Random Oracles: Hash Functions That Hide All Partial Information," in *Advances in Cryptology: Proceedings of Crypto97*, B. S. Kaliski, ed., LNCS 1294, Springer-Verlag (1997), 455–469.

[Common99] "Common Criteria for Information Security Evaluation", version 2.1, CCIMB-99-031, August 1999.

[Davis90] Davis, D. and R. Swick, "Network Security via Private-key Certificates", A*CM Operating Systems Review 24(4)*, Oct 1990, 64-67. Also appeared in Proceedings of U*SENIX UNIX Security III Symposium*, Sept. 1992 14-17, 239-242.

[Davis95] Davis, D., "Kerberos Plus RSA for World Wide Web Security," *In Proceedings of the USENIX Workshop on Electronic Commerce*, July 1995.

[Denning81] Denning, D. and G. Sacco, "Timestamps in Key Distribution," Communications of the ACM, 24(8) August 1981, 533-536.

[DoD85] "Department of Defense Trusted Computer System Evaluation Criteria", DOD 5200.28-STD, December 1985.

[Dumas99] Dumas, D. S. Jacobs, W. Booth and M. Little, "Security Architecture for Intelligent Agent Based Vulnerability Analysis", February, 1999.

[Ephremides99] Ephremides, A., A. Michail and D. Ayyagari, "Hierarchical Scheduling with Route SAFEMITSivity in Wireless Ad-Hoc Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Ephremides00] Ephremides, A. and A. Michail, "Energy-Efficient Routing for Connection-Oriented Traffic in Ad-Hoc Wireless Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Estrin99a], Estrin, D., R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in SAFEMITS Networks", Mobicom '99, August 1999.

[Estrin99b] Estrin, D., "SCADDS Recent Progress", SAFEMITS PI Meeting, Marina Del Rey, October 1999.

[Estrin99c] Estrin, D., "Scalable Coordination Architectures for Deeply Distributed Systems", SAFEMITS KickOff Meeting, Arlington VA, July 1999.

[Falco00] Falco, J., S. Scalera, B. Nelson, N. Srour and A. Filipov, "A Reconfigurable Computing Architecture for MicroSAFEMITSs", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[FIPS140-1] "Security Requirements for Cryptographic Modules", Federal Information Processing Standards Publication, FIPS PUB 140-1, 11 January 1994.

[Ford94] Ford, W., "Computer Communications Security: Principles, Standard Protocols and Techniques", Prentice-Hall, Englewood Cliffs, NJ, 1994.

[Geraniotis00] Geraniotis, E. and J. Thomas, "Antijam Capability of Iterative Multiuser Detection in DS-SSMA", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Govindan99] Govindan, R., T. Faber, J. Heidemann and D. Estrin, "Ad-hoc Smart Environments", In Proceedings of the DARPA/NIST Workshop on Smart Environments, Atlanta, June 1999.

[Imielinski98] Imielinski, T., B.R. Badrinath and J. Freebersyer, "1998 Project Summary: Dataman Project – Information Services for Low-Powered Wireless-Mobile Clients", DARPA ATO Sponsored Research, Rutgers University.

[Joint99] "Joint Technical Architecture", Department of Defense, version 3.0, 15 November 1999.

[Just96] Just, M. and S. Vaudenary, "Authenticated Multi-Party Key," in *Advances in Cryptology – ASIACRYPT'96* (May 1996).

[Kaiser00] Kaiser, W. and G. Pottie, "The Balance Between Local Computation and Communications in Widely Distributed Wireless Embedded Systems", SAFEMITSia Corporation Corporation, 15 January 2000.

[Kelly00a] Kelly, J. and R. Klingeman, "Development of a Notional Man Machine Interface for Interaction between Distributed SAFEMITSs and Handheld Devices", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Kelly00b] Kelly, J., J. Agre, and L. Clare, "On the Need for Data Standardization in Interactive SAFEMITS Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Kommerling99] Kommerling, O. and M. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors", Proceedings of the USENIX Workshop on Smartcard Technology (Smartcard '99), Chicago, Illinois, May 10-11, 1999.

[Kurkoski00] Kurkoski, B., "Design Embedded Systems for Low Power", http://www.edtn.com/embapps/emba002.htm, 2000.

[Laneman00] Laneman, J. and G. Wornell, "Distributed Spatial Diversity Techniques for Improving Mobile Ad-Hoc Network Performance", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Lenstra00] Lenstra, A., and E. Verheul, "Efficient and compact subgroup representation", to be published in the Crypto 2000 Proceedings.

[Lorch95] Lorch, J., "A Complete Picture of the Energy Consumption of a Portable Computer", EECS Master's Thesis, 1995, UC Berkley.

[Lorch98] Lorch, J. and A. Smith, "Software Strategies for Portable Computer Energy Management", IEEE Personal Communications Magazine, vol. 5, no. 3, pp. 60-73, June, 1998.

[Lowe95] Lowe, G., "An Attack on the Needham-Schroeder Public Key Authentication Protocol." *Information Processing Letters*, 56:131-133, 1995.

[Lowe95] Lowe, G. and B. Roscoe, "Using CSP to Detect Errors in the TMN Protocol" *IEEE Transactions on Software Engineering*, 23(10), 659-669, 1997.

[MC68328] "MC68328 Users Manual", Motorola, Preliminary, 6 November 1997.

[McGrew98] McGrew, D., and A., Sherman, "Key establishment in large dynamic groups using one-way function trees," TIS Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD (May 1998).

{Newman94] Newman, B. and T. Ts'o, "Kerberos: an Authentication Service for Computer Networks", *IEEE Communications Magazine*, 32 (September 1994), 33-38.

[Menezes97] Menezes, A., Oorschot, P., and Vanstone, S., "Handbook of Applied Cryptography", CRC Press, New York, 1997.

[Mills00] Mills, D., "Low Energy Communications and Routing for MicroSAFEMITS Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Needham78] Needham, R. and M. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," Communications of the ACM, 21(12) December 1978, 993-999.

[Newman98] Newman, M. and J. Hong, "A Look at Power Consumption and Performance on the 3Com Palm Pilot", UC Berkeley, CS252 Spring 1998.

[OSI88] "OSI Basic Reference Model: Security Architecture", International Standard, ISO 7498-2-1988(E).

[Otway87] Otway D, and O. Rees, "Efficient and Timely Mutual Authentication", *Operating Systems Review*, 21 (1987), 8-10.

[Park94] Park, C. K. Kurosawa, T. Okamoto and S. Tsujii, "On Key Distributation and Authentication in Mobile Radio Networks", in *Advances in Cryptology – EUROCRYPT'93* (May 1993), 461-465.

[Philsar00] Philsar Semiconductor Inc., "PT800 Multi-Purpose RF Transceiver (MuRF)" datasheet.

[RFC2404] C. Madson and R. Glenn, "The use of HMAC-SHA-1-96 within ESP and AH," RFC 2404 (November 1998). ftp://ftp.isi.edu/in-notes/rfc2404.txt.

[Sarneke98] Sarneke, B. and C. Chang, "Ultra-Low Power Communication Logic Circuits for Distributed SAFEMITS Networks", EECS 241, Spring 1998, UC Berkley.

[Schneier96] Schneier, B., "Applied Cryptography", John-Wiley and Sons, New York, 1996.

[Schneier99] Schneier, B., J. Kelsey, D. Whiting, D. Wagner and C. Hall, "Performance Comparison of AES Submissions", v. 2.0, 1 February 1999.

[Singh98] Singh, S., M. Woo and C. Raghavendra, "Power Aware Routing in Mobile Ad-hoc Networks", Mobicom '98, August 98.

[Simmons94] Simmons, G., "Cryptanalysis and Protocol Failures," Communications of the ACM, 37(11) November 1978, 56-65.

[Sirbu97] Sirbu, M. and J. Chuang, "Distributed Authentication in Kerberos Using Public Key Cryptography," Proceedings of the Internet Society 1997 Symposium on Network and Distributed System Security, February, 1997, San Diego, California, IEEE Computer Society (1997).

[Srivastava99] Srivastava, M., "Dynamic SAFEMITS Networks", SAFEMITS PI Meeting, Marina Del Rey, October. 99.

[Stallings99] Stallings, W., "Cryptography and Network Security: Principles and Practice", Prentice-Hall, Upper Saddle River, New Jersey, 1999.

[Stinson95] Stinson, D., "Cryptography Theory and Practice", CRC Press, New York, 1995.

[Tang99] Tang, Z. and J. Garcia-Luna-Aceves, "Hop-Reservation Multiple Access (HRMA) for Ad-Hoc Networks", IEEE INFOCON'99.

[Tassiulas00a] Tassiulas, L and J. Chang, "Maximum Lifetime Routing in Wireless SAFEMITS Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Tassiulas00b] Tassiulas, L and J. Chang, "Energy Conserving Routing in Wireless Ad-Hoc Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Tate89] Tatebayashi, M., N. Matsuzaki and D. Newman. "Key Distribution Protocol for Digital Mobile Communications Systems," in *Advances in Cryptology: Proceedings of Crypto89*, G. Brassard, ed., LNCS 435, Springer-Verlag (1989), 324-334.

[Tayong00] Tayong, H., et al, "A Reduced Complexity Detector for Fast Frequency Hopping", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Tung00a] Tung, B., et al., R., "Public Key Cryptography for Initial Authentication in Kerberos," Internet Draft (work in progress), draft-ietf-cat-kerberos-pk-init-12.txt, July 15, 2000.

[Tung00b] Tung, B., et al., R., "Public Key Cryptography for Cross-Realm Authentication in Kerberos," Internet Draft (work in progress), draft-ietf-cat-kerberos-pk-cross-4.txt, July 15, 2000.

[vanHook99] vanHook, D., "Dynamic Declarative Networking", SAFEMITS PI Meeting, Marina Del Rey, October. 99.

[van Oorschot92] van Oorschot, "A Alternative Explanation of Two BAN-logic "Failures"," Advances in Cryptology: Proceedings of – *EUROCRYPT'93*, LNCS 765, Springer-Verlag (1994), 443-447.

[Wang00a] Wang, A., W. Heinzelman and A. Chandrakasan, "Energy-Scalable Protocols for Battery-Operated MicroSAFEMITS Networks", Proceedings of the ARL Federated Laboratory 4th Annual Symposium, 21-23 March 2000, College Park, MD.

[Wang00b] Wang, M., "Control Scheme Gives Power Tune-Up", Electronic Engineering Times, 1 May 2000, p. 92.

[WINS NG00] "WINS NG Power Usage Specification: WINS NG 1.0", SAFEMITSia Corporation, January 2000.

[Wittman99] Wittman, A., "Feeding Moore's Law", Network Computing Magazine, Issue 1026, 27 December 1999.

Acronyms

ACL	Access Control List
AES	Advanced Encryption Standard
AJ	Anti-Jam
ARL	Army Research Laboratory
ASIC	Application-Specific Integrated Chip
ATIRP	Advanced Telecommunications & Information Distribution Research Program
BD	Burmester-Desmedt
BPSK	Binary Phase Shift Keying
C	Capacitance
C^2	Command and Control
CBC	Cipher Block Chaining
CDMA	Code Division Multiple Access
CFB	Cipher Feedback
CKM	Cryptographic Key Management
COMSEC	Communication Security
CONOPS	Concept of Operations
COTS	Commercial Off The Shelf
CRC	Cyclic Redundancy Code
CRL	Certificate Revocation List
DARPA	Defense Advanced Research Project Agency
DF	Direction Finding
DoD	Department of Defense
DoS	Denial of Service
DS	Direct Sequence
DSN	Distributed SAFEMITS Network
ECB	Electronic Codebook
EM	Electromagnetic
EMF	Electromagnetic Frequency
EPROM	Erasable Programmable Read Only Memory
EEPROM	Electrically Erasable Programmable Read Only Memory
f	clock frequency
FHMA	Frequency Hopping Multiple Access
FIPS	Federal Information Processing Standard
FSRS	Functional Security Requirement Specification
GDH	Group Diffie-Hellman
GPS	Global Positioning System
IETF	Internet Engineering Task Force
IV	Initialization Vector
J	Joule
LAN	Local Area Network
LPD	Low Probability of Detection
LPI	Low Probability of Interception
MAC	Medium Access Control
MANET	Mobile Ad-Hoc Networking
MEMS	Microelectromechanical Systems
MIPS	Million Instructions Per Second
MOUT	Military Operations on Urbanized Terrain
NAI	Network Associates Incorporated

NG	Next Generation
NSA	National Security Agency
QoS	Quality of Service
P	Power
PDA	Personal Digital Assistant
PLGR	Precision Lightweight GPS Receiver
RAM	Random Access Memory
RF	Radio Frequency
RFC	Request For Comments
ROM	Read Only Memory
SAFEMITS	SAFEMITS Information Technology
SHA	Secure Hash Algorithm
SFA	Security Fault Analysis
TCP	Transmission Control Protocol
TEMPEST	Thermal, Electro-Magnetic and Physical Equipment Stress Testing
USNO	United States Naval Observatory
UTC	Coordinated Universal Time
V	Voltage
W	Watt
WINS	Wireless Integrated Network SAFEMITS